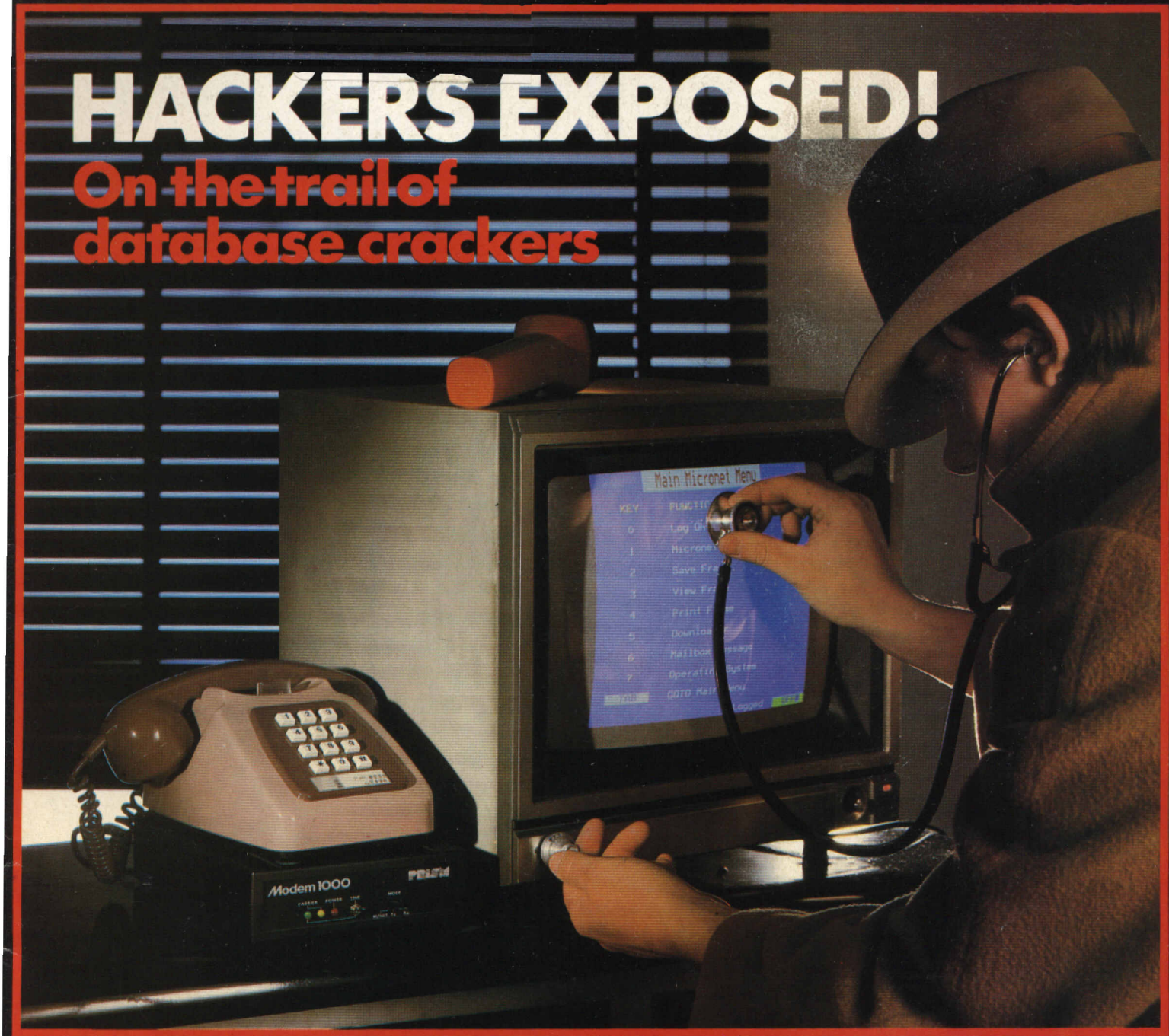
THE BEST SELLING COMPUTER PROJECTS MAGAZINE MARCH 1985

ELECTRONICS & COMPUTING

AN EMAP PUBLICATION

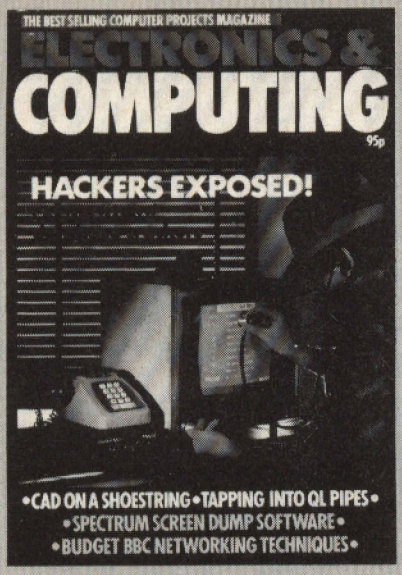
USA $2.95
Germany D6.00
Singapore S$4.95
95p



HACKERS EXPOSED!

On the trail of database crackers

- CAD ON A SHOESTRING
- TAPPING INTO QL PIPES
- SPECTRUM SCREEN DUMP SOFTWARE
- BUDGET BBC NETWORKING TECHNIQUES



COVER PHOTO BY ROB BRIMSON

ELECTRONICS & COMPUTING Contents

Vol. 5 Issue 3

Electronics & Computing Monthly
Priory Court, 30-32 Farringdon Lane,
London, EC1R 3AU

Editorial 01-251-6222
Editor Gary Evans
Deputy Editor William Owen
Production Editor Liz Gregory

Advertising 01-251-6222
Advertisement Manager Anthony Herman
Advertising Production Serena Hadley

Production 01-251-6222
Art Editor Jeremy Webb
Make-up Time Graphics
Publisher Terry Pratt

Distribution
EMAP National Publications

Published by
EMAP Business and Computer Publications

Printed by
Riverside Press, England

Subscriptions
Subscriptions and back issues please telephone 01-251 6222 for details

Electronics & Computing Monthly is normally published on the 13th day of each month.

PROJECTS

BBC circuit drafter COVER FEATURE 14
Turns the BBC micro into a low cost CAD system capable of producing high quality circuit diagrams complete with component annotation.

Spectrum screen dump COVER FEATURE 20
Allows graphic displays to be dumped to any Epson compatible printer.

Low cost networking COVER FEATURE 34
Paul Beverley with details of how a number of BBC micros may be connected in a network system without the expense often associated with such an exercise.

Disk catalouger 38
A utility program that allows all the disks in your collection of software to be catalouged into one comprehensive file. You'll probably discover some forgotten program gems after giving your disks the treatment.

Slot car timing 49
An extension of last month's software allows two cars to be timed. The technique adopted means that machine code programming can be avoided.

FEATURES

Tapping into QL pipes COVER FEATURE 25
Pipes are a means of passing data between programs running on the QL. Mastering their operation is a further step toward unleashing the full power of the QL.

Random access 42
Adam Denning continues the major *E&CM* software project that aims to provide the BBC micro with a full random access filing system.

Hackers exposed COVER FEATURE 46
We have some advice for database managers from our own super-hacker. Following his advice could cut down on the number of unauthorised accesses to commercial database operations.

Ribbon economiser 52
A simple modification to a printer ribbon cartridge can make your ribbon last and last and . . .

Communications column 62
This month Ben Knox has produced an extensive list of bulletin board and PSS information.

REVIEWS

CMS 6809 card 60
A 6809 processor card for the beeb that opens up the world of the powerful FLEX operating system.

Editorial 10
News 10
Next Month 28
Book service 41
PCB service 63

And within the pages of *Your Robot* starts page 55

News from the world of robotics plus details of the latest stage in our LEGO robot building project.

WORDWISE +

Following the success of Wordwise, Computer Concepts wordprocessing package for the BBC micro, the company have launched an upgrade to enhance the capabilities of the 8K program. Wordwise Plus will occupy 16K and features several features which improve upon its predecessor.

The main difference between the two is the way in which the Plus program organises storage facilities. Because any mixture of programs and text may be stored in up to 10 segments, it will allow the user to store and edit anything up to 11 different items at any one time. One of the segments is reserved as a kind of notebook which may be accessed very easily from the document being worked on, so that users can cross-reference very quickly.

Computer Concepts claim that the loading and saving of disks has been improved ten fold on the older version. A useful facility is the giving of a warning when text is in danger of being overwritten. In addition 6502 2nd processor users will be able to obtain a disk version which will allow extra memory storage for text.

Packages which come ready supplied with Wordwise Plus include programs for mailmerging, multiple copying, two column printing and index generating. Wordwise Pluse costs £56.35 including VAT and an update for Wordwise owners is available for £19.55. *Computer Concepts, Gaddesden Place, Hemel Hempstead, Herts HP2 6EX. Tel: 0442 63933*

Make a date

Digital Lab and Desk Diary are two new packages for the Commodore 64 from Associated Services Ltd. The former allows the designing of logic circuits and, once these are completed, provides facilities for the checking of all logic gates and links. These circuits may be saved onto disk and, in addition, Digital Lab allows the formatting, copying, deleting and renaming of data files.

Desk Diary is provided on two disks and provides a useful method of entering and protecting diary information. Thus appointments and engagements up until 31 March 1992 may be recorded and any other information, for example financial data, may be added and updated. The software also allows users to leave messages for other diary users and accesses the dates of any month from the years 1983 to 2057.

Both packages retail for £29.95 inc. VAT each and may be obtained from *Associated Services, 23 Chesham St, London SW1X 8NQ.* Please allow £1.00 extra for package and postage.

A BROAD VIEW

In the jargon of the newstrade, *Electronics and Computing* is known as a horizontal magazine. This term reflects the fact that, while *E&CM* is a special interest computer magazine, it casts its editorial net over a wide range of computer interests. This fact sets it apart from the other group of computer magazines, the vertical titles. These magazines, often identified by the incorporation of the word 'user' in their title, restrict their coverage to just one type of computer, or to the range of machines produced by a particular manufacturer.

In taking the broad view of the microcomputer industry every month, *E&CM* has the problem of squeezing into a limited number of editorial pages all the material that we would like to publish. It's a well established publishing fact, indeed a fact of life, that we will not be able to please all of our readers all of the time but we hope to satisfy the requirements of the majority.

Over recent months we have been open to the criticism that the magazine has concentrated on the BBC micro to the exclusion of other computers. A study of the editorial over the past six months will show that, while we have carried quite a number of BBC orientated features, we have not forgotten the Spectrum users nor indeed the owners of other popular machines. In addition we aim to publish regular features that will appeal to readers no matter what specific computer they own.

In the coming months we will endeavour to include porjects and features aimed at a broad range of hardware. In particular Spectrum owners can look forward to a major series of projects later on in the year. The Spectrum computer does itself no favours when it comes to interfacing to the outside world as Sinclair have not seen fit to incorporate much by way of user interfaces within the computer's specification. To overcome this problem we have come up with a general purpose expansion system which will enable the computer to interface with a wide range of projects that we have in the design pipeline.

We will also be taking an interest in the Commodore 64 and Dragon computers. The latter, while no longer in current UK production, has a large number of users in the field who at present are ill-served by the computer press.

Other topics that we shall be covering in the coming year include a close monitoring of the introduction of the 16-bit computer. Both Atari and Commodore have impressive hardware ready for launch later on in the year when these organisations will join the established Sinclair QL in opening up the power of the 16-bit processor. Another field which will see an increased presence within *E&CM* is the area of inter-micro communication and the subject of databases. The opportunities opened up by services such as Prestel/Micronet and Commodore's Compunet are just beginning to be realised by commercial concerns. *E&CM* will be monitoring developments on this front from the micro users point of view.

The next year will be an interesting one for microcomputer owners. We hope you will continue to keep in touch by way of the coverage given in *Electronics and Computing.*

GARY EVANS

Spectrum storage solution

A 3.5" Spectrum disk drive developed jointly by Hitachi and Servicon Dynamics will be available in the shops by April.

The unit carries a 12 month warranty. It is powered from the Spectrum and has a storage capacity of 128K with an expansion socket for copying and advanced programming which takes the capacity up to 256K. The unit also has an RS423 interface with 'D' type connector for dial-up services, and an RGB socket for colour monitor attachment.

Sean Mayo, Servicon's marketing manager, claims that the unit 'is the most cost effective, reliable solution to fast loading; reading and writing 64K in 4-8 seconds'.

The Crescent 128 costs £99.95 plus VAT and is designed as an expansion unit to the Crescent 128 which costs £129.95. The two units together give a capacity of 256K.

Soft Release

Lots more QL software of the assembler variety coming out, but little in the way of applications or games though Sinclair promise QL versions of the Imagine 'Megagames' in the near future.

QCODE is a superBASIC screen editor (called QED) and 68000 assembler. It saves and assembles the source code to produce the 'object program' once the source is written using the editor. If the source program fails to assemble because of syntax errors, or the object program fails to execute correctly because of a logical error the source program can then be read back into memory and re-edited. For those familiar with the 68008 instruction set only.

Second QL assembler in a month is the Macro Assembler from Adder Publishing (£29.95). This includes 68000 assembler, editor and debugger. Assembler source text, symbol table and assembled program are kept in memory all the time so, with small programs at least, no interaction with the microdrive is necessary. Longer programs can be split into files, which, say Adder, encourages a modular program design for the assembly of very large programs.

For the BBC micro, Tomorrow's Dream Software have launched Titan, a debugging monitor facility which allows breakpoints and signle stepping through ROM and RAM, is fully relocatable, has intelligent memory mover, single stepping through graphics routines, and intelligent masked string search routines. Titan is available on tape at £10 and on disk £12. Tomorrow's Dream say that this is only the first of number of utilities for the BBC micro, and that Titan will be available on microdrive for the 48K Spectrum early in 1985.

QL disk

The manufacturer of a new disk drive expansion system for the Sinclair QL claims that all current software will run (including the Psion packages) and that it needs no expansion board to join with the QL.

The software for the 'Computamate Q-Disk' was written by Tony Tebby, the author of QDOS. The interface, which is priced at £149, contains on-board ROM with QDOS device driver, can cope with double density 3.5" or 5.25" discs, has an access time of 125 milliseconds (the microdrive is 3.5 secs minimum) and handles up to 512 bytes per sector on 40 or 80 tracks, single or double sided disks.

The ROM contains what are described as all necessary utilities for disk and file handling. Computamate Data Products are on 0782 811711.

NEWS NEWS NEWS NEWS

Pictured below is the lucky winner of the competition run in conjunction with Twilstar Computers. The £500 prize went to Mr. L. Tomlin (centre), it was his completed *E&CM* questionnaire that was selected from all those received at our offices. Also pictured are Gary Evans (left) *E&CM's* Editor and Mr. K. Dhesi of Twilstar.

Microvitec, the monitor people, have launched a new touchscreen which they hope will have an impact on the educational field. The company feel that, as no keyboard skills are required when using the peripheral, there is scope for very young children to get to know more about computers as well as perhaps disabled people. Using a network of infra-red beams, the screen may react simply under the touch of a finger of stylus and this does not affect the quality of the screen display. The software provided includes nine demonstration programs and this has been produced in conjunction with the MEP. The Touchscreen retails at £210 plus VAT and further information may be obtained from *Microvitec, Futures Way, Bolling Road, Bradford, BD4 7TU; Telephone: 0274 390011.*

All change

Despite carrying out a thorough search when they commenced trading over a year ago, Computer Link have now found that there is a foreign company with the same name. Although this company do not supply the same goods as the UK Computer Link, they have strongly objected to the use of their registered name and so the British firm have agreed to change it to RSD Connections Ltd. The company may still be found at *PO Box 1, Ware, Herts* and still have the same telephone number – 0920 5285.

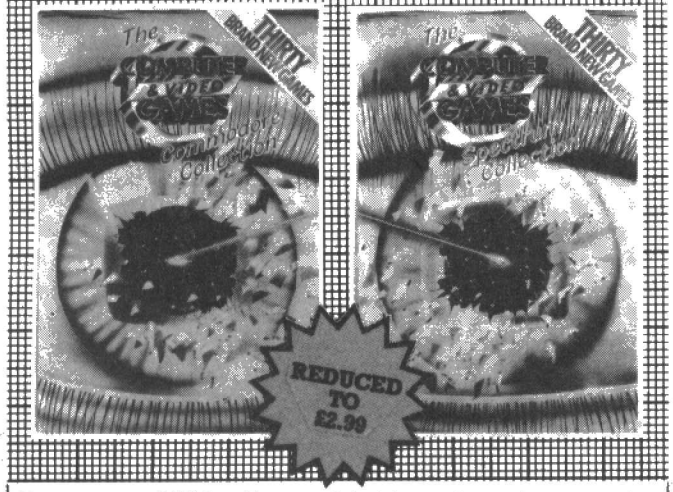
HEART RATE MONITOR USERS PLEASE NOTE

Our February issue featured a free cover-mounted gift which showed readers how to build a heart rate monitor. This item failed to stipulate that, with projects of this nature, any attempt to operate the circuit from a power supply unit could prove to be highly dangerous. It should be noted, therefore, that this should only be powered by batteries and on no account must it be used with a mains PSU.

Modems are ever coming down in price. The recently introduced Unicom modem for example has a price tag of only £49.95 (plus VAT) and features full and half duplex, auto dial, and auto baud rate sensor. Software is available for the BBC micro and expected soon for the Spectrum and Amstrad CPC464, and then Dragon and Tandy colour computers.

Commodore Collection — COMPUTER & VIDEO GAMES — Spectrum Collection

ISBN No: 185096002X ISBN No: 185096003X



Computer and Video Games, Britain's most popular computer games monthly, bring you two exciting new books for the Spectrum and Commodore 64.

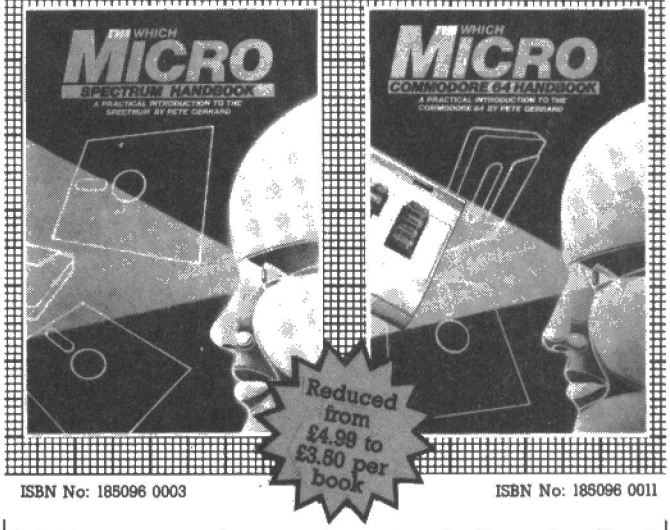
Each 100-page book contains up to 30 **brand new** games checked and prepared by **Computer and Video Game's** regular contributors.

At £2.99 reduced from £4.99 - the best value on the market at all good bookshops or send to EMAP Books, Bushfield House, Orton Centre, Peterborough PE2 0UW.

WHICH MICRO

COMMODORE 64 HANDBOOK

SPECTRUM HANDBOOK



ISBN No: 185096 0003 ISBN No: 185096 0011

Britain's two most popular personal computers, the Commodore 64 and Spectrum are covered from basic to semi-expert in **Which Micro Magazine's Handbooks.**

Author Pete Gerrard, a regular columnist for Which Micro, has put together an accurate and practical guide to both computers, reduced from £4.99 to £3.50 per book.

Many programs are included and both 160-page books are spiral bound for easy use and are available in all good bookshops from November or direct through your letterbox by sending to EMAP Books, Bushfield House, Orton Centre, Peterborough PE2 0UW.

CIRCUIT DRAFTER

While not offering all the facilities of a super expensive icon caden CAD system, Nick Holmes' program for the BBC micro can produce high quality circuit diagrams.

In industry, the task of producing clear, easy to comprehend circuit diagrams, is today the province of multi-thousand pound CAD drafting systems. The days of drawing circuits out on paper has long been consigned to the pages of history. For the home user though, there is often no alternative to the time honoured method of producing diagrams as the latest mouse-driven, icon-laden CAD systems have associated with them a price tag that is more than most of us would pay for a house, let alone another add-on for the BBC micro.

The software described here, however, offers owners of the BBC computer a chance to explore the world of CAD drafting, and uses an easy to operate and (fairly) self explanatory all BASIC that provides a surprisingly sophisticated output as the sample printouts show.

Quick on the draw

The main program is shown in **Listing 2** while **Listing 1** shows a custom written screen dump routine for Epson FX-80 and FX-100 printers and their clones. The printer dump routine is listed in Assembler and should be assembled into machine code, by typing it in and running it, before dumping it to tape after the main program: use the command "*SAVE PRNT A00 A82' for a disk system and "*SAVE FNHI E88 E82' for a tape system. This means that when the main program is CHAINed it will load the printer driver code automatically at line 40 and the user will not have to think about it again once the program is first set up. The drafting program loads the print routine which is called when the P key is hit.

Two important notes: firstly, if your BBC micro is fitted with a ROM-driven disk system, the drafting program may not fit into the area normally reserved for a BASIC program and it may be necessary to reset 'PAGE' to around &1500 before loading: secondly, with a tape system the machine code screen dump routine will occupy PAGE E and it will be necessary to set PAGE to &F00 before loading.

In operation

Upon startup, the graphics screen is blank except for the cursor in the middle. Note that this is not the usual graphics cursor and is moved with the 'normal' cursor control keys without marking the paper. Lines are drawn by the four keys adjacent to the cursor key cluster: '\' draws upwards, '_' to the right, ']' downwards and '[' to the left. Each press of a key draws a line that is two pixels long and holding a key down will draw a further two pixels at each key 'bounce'. This makes drawing in fine detail with single key presses as easy as drawing lines.

The commands used to draw the predefined components are listed in a text window along the lower edge of the

LISTING 1

```
100SBYTE=&FFF4:OSWRCH=&FFEE:STR=&77:SCR=&75:COP=&72:ROW=&71
  :CLM=&70:PT RBYTE=&85
   20 FOR Z%=0TO3 STEP 3
   30 P%=&A00:REM P%=&E00 FOR TAPE
   40 [OPT Z%
   50 LDA £27
   60 JSR WRPTR
   70 LDA £65
   80 JSR WRPTR
   90 LDA £8
  100 JSR WRPTR
  110 LDA £&30
  120 STA SCR+1
  130 LDA £0
  140 STA SCR
  150 LDA £80
  160 STA CLM
  170 LDA £32
  180 STA ROW
  190 .SEND
  200 LDA CLM
  210 STA COP
  220 LDA £27
  230 JSR WRPTR
  240 LDA £76
  250 JSR WRPTR
  260 LDA £128
  270 JSR WRPTR
  280 LDA £2
  290 JSR WRPTR
  300 .LP
  310 LDY £7
  320 .LOP
  330 LDA (SCR),Y
  340 STA STR,Y
  350 DEY
  360 BPL LOP
  370 LDY £8
  380 .LOOP
  390 LDX £7
  400 .LOOOP
  410 ASL STR,X
  420 ROR A
  430 DEX
  440 BPL LOOOP
  450 JSR WRPTR
  460 DEY
  470 BNE LOOP
  480 LDA SCR
  490 CLC
  500 ADC £8
  510 STA SCR
  520 BCC CLEER
  530 INC SCR+1
  540 .CLEER
  550 DEC COP
  560 BNE LP
  570 LDA £&0A
  580 JSR WRPTR
  590 DEC ROW
  600 BNE SEND
  610 LDA £27
  620 JSR WRPTR
  630 LDA £50
  640 JSR WRPTR
  650 RTS
  660 .WRPTR
  670 STA PTRBYTE
  680 LDA £1
  690 JSR OSWRCH
  700 LDA PTRBYTE
  710 JSR OSWRCH
  720 RTS
  725 ]
  730 NEXT Z%
```

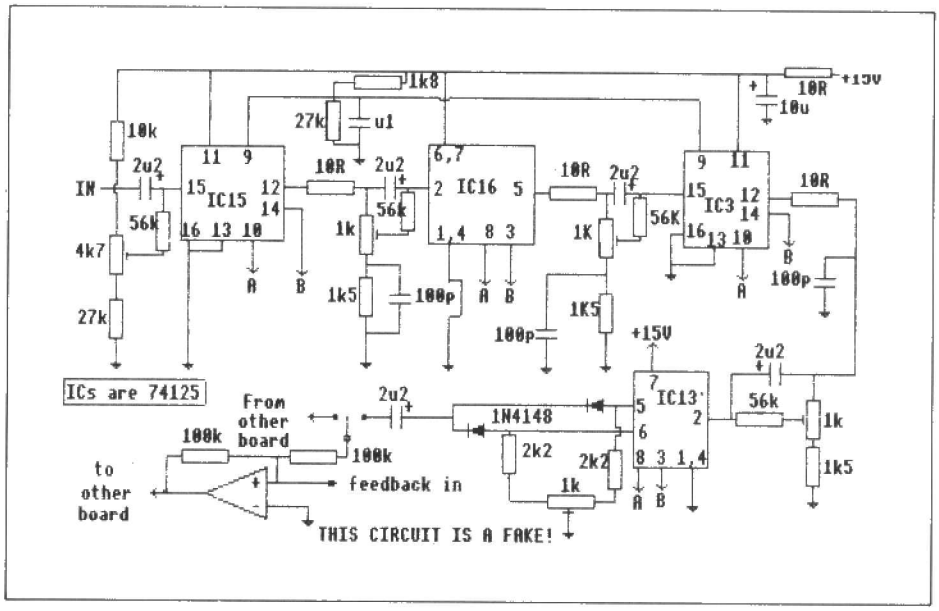
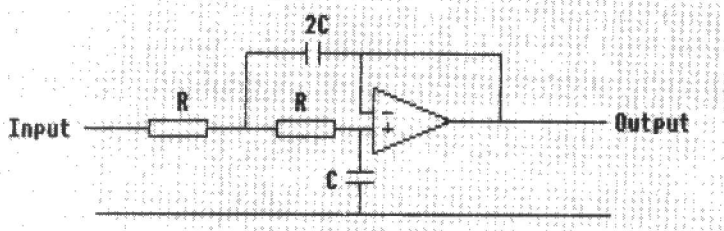


Figure 1. Typical program output.

screen. The components are Resistors, Capacitors (unbiased), Diodes, Transistors and Operational Amplifiers. Passive components are drawn from the current cursor position in whichever direction is specified (up, down, left or right). Electrolytic capacitors can be drawn by including a + beside one plate. Active components may be either left or right although transistors may be either NPN or PNP. The Arrow command may be used with the available component set to make up potentiometers, LEDs, LDRs, etc. and in conjunction with the line drawing commands to form JFETs and MOSFETs: as an added bonus,

LISTING 2

```
   10 MODE0
   15 VDU19,1,0,0,0,0:VDU19,128,6,0,0,0
   20 DIM COD 30
   30 VDU28,0,31,79,29:REM SET UP TEXT
WINDOW.
   32 VDU24,0;32;1279;1023;
   35 CLS
   40 *LOAD PRNT
   50 VDU23,135,0,0,0,0,0,0,0,0
   60 *DIR C
  100 *FX4,1
  105 MOVE 600,500:FLAG=0
  106 PROCMENU
  109 ON ERROR GOTO9900
  110 REPEAT
  115 PROCCURSOR
  120 KEY=GET
  125 PROCCURSOR
  130 IF KEY=&88 THEN PLOT0,-4,0:GOTO200
  132 IF KEY=&89 THEN PLOT0,4,0:GOTO200
  134 IF KEY=&8B THEN PLOT0,0,4:GOTO200
  136 IF KEY=&8A THEN PLOT0,0,-4:GOTO200
  140 IF KEY=&5B THEN PLOT1,-4,0:GOTO200
  142 IF KEY=&5F THEN PLOT1,4,0:GOTO200
  144 IF KEY=&5C THEN PLOT1,0,4:GOTO200
  146 IF KEY=&5D THEN PLOT1,0,-4:GOTO200
  148 IF KEY=&7B THEN PLOT3,-4,0:GOTO200
  150 IF KEY=&60 THEN PLOT3,4,0:GOTO200
  152 IF KEY=&7C THEN PLOT3,0,4:GOTO200
  154 IF KEY=&7D THEN PLOT3,0,-4:GOTO200
  155 IF KEY=&0D THEN PROCMENU:GOTO200
  156 IF KEY=&72 OR KEY=&52 THEN
CLS:PRINT"RESISTOR:":PROCDIRECTION:PROCRESISTOR(DIR):PROCMENU:GOTO200
  158 IF KEY=&63 OR KEY=&43 THEN
CLS:PRINT"CAPACITOR:":PROCDIRECTION:PROCCAPACITOR(DIR):PROCMENU:GOTO20
0
  160 IF(KEY=&74 OR KEY=&54)AND FLAG=0 THEN CLS:PRINT"TRANSISTOR: MOVE
THE CURSOR TO THE LOCATION OF THE 'BASE' TERMINAL
       AND PRESS 'T' AGAIN":LET FLAG=1:GOTO200
  161 IF(KEY=&74 OR KEY=&54)AND FLAG=1 THEN CLS:PROCTRANSISTOR:LET
FLAG=0:PROCMENU:GOTO200
  165 IF (KEY=&4B OR KEY=&6B) AND FLAG=0 THEN CLS:PRINT"CONFIRM WIPE
WHOLE DISPLAY BY PRESSING 'K' AGAIN":FLAG=2:GOTO200
  166 IF (KEY=&4B OR KEY=&6B) AND FLAG=2 THEN CLG:PROCMENU:GOTO200
  167 IF KEY=&57 OR KEY=&77 THEN PRINT"WRITING TEXT AT CURRENT CURSOR
POSITION...";'"PRESS <COPY> TO END TEXT":PROCTEXT:PROCMENU:GOTO200
  172 IF KEY=&64 OR KEY=&44 THEN
PRINT"DIODE...":PROCDIRECTION:PROCDIODE(DIR):PROCMENU:GOTO200
  174 IF KEY=&42 OR KEY=&62 THEN PRINT"THE BATTERY WILL APPEAR WITH
THE NEGATIVE TERMINAL AT THE
CURSOR...":PROCDIRECTION:PROCBATTERY(DIR):PROCMENU:GOTO200
  176 IF KEY=&50 OR KEY=&70 THEN CLS:PROCPRINT:PROCMENU:GOTO200
  178 IF KEY=&53 OR KEY=&73 THEN PROCSAVE:PROCMENU:GOTO200
  180 IF KEY=&6C OR KEY=&4C THEN PROCLOAD:PROCMENU:GOTO200
  182 IF (KEY=&45 OR KEY=&65) AND FLAG=0 THEN PRINT"CURRENT CURSOR
POSITION IS BOTTOM LEFT CORNER OF AREA TO BE
ERASED":POSX1=?&310+256*?&311:POSY1=?&312+?&313*256:PRINT"MOVE CURSOR
TO TOP RIGHT OF AREA AND PRESS 'E' AGAIN":FLAG=3:GOTO200
  183 IF (KEY=&45 OR KEY=&65) AND FLAG=3 THEN
PROCERASE:PROCMENU:GOTO200
  185 IF KEY=&41 OR KEY=&61 THEN PRINT"ARROW: The current cursor
position is the tip of the
arrow...":PROCDIRECTION:PROCARROW(DIR):PROCMENU:GOTO200
  187 IF (KEY=&4F OR KEY=&6F) AND FLAG=0 THEN PRINT"OPERATIONAL
AMPLIFIER";'"PLACE CURSOR MIDWAY BETWEEN POSITIVE AND NEGATIVE INPUTS,
THEN PRESS 'O' AGAIN":FLAG=4:GOTO200
  188 IF (KEY=&4F OR KEY=&6F) AND FLAG=4 THEN
PROCOPAMP:PROCMENU:GOTO200
  200 UNTIL FALSE
 1000 DEFPROCMENU
 1005 FLAG=0
 1010 CLS:PRINT"Resistor..Capacitor..Write
text..Transistor..Arrow..Diode..Battery..Op-amp.....Klear
screen..Save/Load to tape/disc..Print screen..Erase area...":ENDPROC
 2000 DEFPROCCURSOR
 2010 PLOT0,-16,-16
 2020 PLOT2,32,32
 2030 PLOT0,-32,0
 2040 PLOT2,32,-32
 2050 PLOT0,-16,16
 2060 ENDPROC
 2200 DEFPROCDIRECTION
 2210 PRINT "Up, Down, Left, or Right?"
 2220 A$=GET$
 2230 IF A$="R" OR A$="r" THEN DIR=1:ENDPROC
 2240 IF A$="L" OR A$="l" THEN DIR=2:ENDPROC
 2250 IF A$="D" OR A$="d" THEN DIR=3:ENDPROC
 2260 IF A$="U" OR A$="u" THEN DIR=4:ENDPROC
 2270 GOTO2210
 8500 DEFPROCDIODE(X)
 8510 ON X GOSUB 8530,8540,8550,8560
 8520 ENDPROC
 8530
PLOT1,8,0:PLOT0,0,12:PLOT1,0,-24:PLOT81,24,12:PLOT0,0,12:PLOT1,0,-24:P
LOT0,0,12:PLOT1,8,0:RETURN
 8540
PLOT1,-8,0:PLOT0,0,12:PLOT1,0,-24:PLOT81,-24,12:PLOT0,0,12:PLOT1,0,-24
:PLOT0,0,12:PLOT1,-8,0:RETURN
 8550
PLOT1,0,-8:PLOT0,12,0:PLOT1,-24,0:PLOT81,12,-24:PLOT0,12,0:PLOT1,-24,0
:PLOT0,12,0:PLOT1,0,-8:RETURN
 8560
PLOT1,0,8:PLOT0,12,0:PLOT1,-24,0:PLOT81,12,24:PLOT0,12,0:PLOT1,-24,0:P
LOT0,12,0:PLOT1,0,8:RETURN
 8600 DEFPROCBATTERY(X)
 8610 ON X GOSUB 8630,8640,8650,8660
 8620 ENDPROC
 8630
PLOT1,8,0:PLOT0,0,-12:PLOT1,0,24:PLOT0,16,12:PLOT1,0,-48:PLOT0,0,24:PL
OT1,8,0:RETURN
 8640
PLOT1,-8,0:PLOT0,0,-12:PLOT1,0,24:PLOT0,-16,12:PLOT1,0,-48:PLOT0,0,24:
PLOT1,-8,0:RETURN
 8650
PLOT1,0,-8:PLOT0,12,0:PLOT1,-24,0:PLOT0,-12,-16:PLOT1,48,0:PLOT0,-24,0
:PLOT1,0,-8:RETURN
 8660
PLOT1,0,8:PLOT0,12,0:PLOT1,-24,0:PLOT0,-12,16:PLOT1,48,0:PLOT0,-24,0:P
LOT1,0,8:RETURN
 8700 DEFPROCPRINT
 8710 *FX6,0
 8720 *FX5,1
 8722 VDU2
 8725 CALL&A00:REM FOR TAPE, CALL &E00
 8727 VDU3
 8730 ENDPROC
 8750 DEFPROCSAVE
 8760 CLS:INPUT"ENTER FILENAME";B$
 8765 IF B$="" THEN ENDPROC
 8770 $COD="SAVE "+B$+" 3000 7FFF"
 8780 X%=COD MOD 256:Y%=COD DIV 256
 8790 CALL&FFF7
 8800 ENDPROC
 8850 DEFPROCLOAD
 8860 CLS:INPUT"ENTER FILENAME";B$
 8870 IF B$="" THEN ENDPROC
 8880 $COD="LOAD "+B$
 8890 X%=COD MOD 256:Y%=COD DIV 256
 8900 CALL&FFF7:ENDPROC
 9000 DEFPROCCAPACITOR(DIR)
 9010 ON DIR GOSUB 9050,9080,9110,9140
 9020 ENDPROC
 9050
PLOT1,16,0:PLOT0,0,20:PLOT1,0,-40:PLOT0,16,40:PLOT1,0,-40:PLOT0,0,20:P
LOT1,16,0
 9060 RETURN
 9080 PLOT0,-48,0:GOSUB9050:PLOT0,-48,0:RETURN
 9110
PLOT1,0,-16:PLOT0,16,0:PLOT1,-32,0:PLOT0,32,-16:PLOT1,-32,0:PLOT0,16,0
:PLOT1,0,-16
 9120 RETURN
 9140 PLOT0,0,48:GOSUB9110:PLOT0,0,48:RETURN
 9150 RETURN
 9200 DEFPROCTRANSISTOR
 9210 PRINT "Npn or Pnp?":TYPE$=GET$:IF TYPE$<>"N" AND TYPE$<>"P" THEN
GOTO9210
 9220 PRINT"Left or Right?":DIR$=GET$:IFDIR$<>"L" AND DIR$<>"R" THEN
GOTO9220
 9230 IF TYPE$="N" AND DIR$="L" THEN PROCTRA:ENDPROC
 9235 IF TYPE$="N" AND DIR$="R" THEN PROCTRB:ENDPROC
 9240 IF TYPE$="P" AND DIR$="L" THEN PROCTRC:ENDPROC
 9245 IF TYPE$="P" AND DIR$="R" THEN PROCTRD:ENDPROC
 9250 ENDPROC
 9260 DEFPROCTRA
 9270
PLOT0,0,-24:PLOT1,0,48:PLOT0,-32,0:PLOT1,32,-24:PLOT1,-32,-24:PLOT0,12
,0:PLOT1,-12,0:PLOT1,0,12:PLOT0,0,-12:ENDPROC
 9280 DEFPROCTRB
 9290
PLOT0,0,-24:PLOT1,0,48:PLOT0,32,0:PLOT1,-32,-24:PLOT1,32,-24:PLOT0,-12
,0:PLOT1,12,0:PLOT1,0,12:PLOT0,0,-12:ENDPROC
 9300 DEFPROCTRC
 9310
PLOT0,0,24:PLOT1,0,-48:PLOT0,-32,0:PLOT1,32,24:PLOT1,-32,24:PLOT0,24,-
8:PLOT1,0,-12:PLOT1,-12,0:PLOT0,-12,20:ENDPROC
 9320 DEFPROCTRD
 9330
PLOT0,0,24:PLOT1,0,-48:PLOT0,32,0:PLOT1,-32,24:PLOT1,32,24:PLOT0,-24,-
8:PLOT1,0,-12:PLOT1,12,0:PLOT0,12,20:ENDPROC
 9400 DEFPROCTEXT
 9410 VDU5
 9415 REPEAT
 9420 A$=GET$:PRINTA$;:UNTIL ASC(A$)=&87:VDU4:ENDPROC
 9430
 9500 DEFPROCERASE
 9510 POSX2=?&310+?&311*256:POSY2=?&312+?&313*256
 9520
PLOT4,POSX1,POSY1:PLOT4,POSX1,POSY2:PLOT87,POSX2,POSY2:PLOT4,POSX2,POS
Y1:PLOT87,POSX1,POSY1
 9530 ENDPROC
 9600 DEFPROCARROW(X)
 9610 ON X GOSUB 9630,9640,9650,9660
 9620 ENDPROC
 9630 PLOT1,-32,0:PLOT0,24,8:PLOT1,8,-8:PLOT1,-8,-8:RETURN
 9640 PLOT1,32,0:PLOT0,-24,8:PLOT1,-8,-8:PLOT1,8,-8:RETURN
 9650 PLOT1,0,32:PLOT0,8,-24:PLOT1,-8,-8:PLOT1,-8,8:RETURN
 9660 PLOT1,0,-32:PLOT0,8,24:PLOT1,-8,8:PLOT1,-8,-8:RETURN
 9700 DEFPROCOPAMP
 9705 PLOT0,0,48:PLOT1,0,-96
 9710 PRINT"LEFT OR RIGHT?":A=GET:IF A=&4C OR A=&6C THEN GOSUB 9730
ELSE GOSUB9740
 9720 ENDPROC
 9730 PLOT1,-96,48:PLOT1,96,48:PLOT0,-96,-48:RETURN
 9740 PLOT1,96,48:PLOT1,-96,48:PLOT0,96,-48:RETURN
 9800 *DIR C
 9805 PROCCURSOR:GOTO106
 9900 PRINT"IF YOU WANT TO RE-ENTER THE PROGRAM WITHOUT CLEARING THE
SCREEN,TYPE 'GOTO9800'";
 9905 *DIR$
 9910 STOP
 9999 DEFPROCRESISTOR(X)
10020ON X GOSUB 10030,10070,10110,10140
10025 ENDPROC
10030
PLOT1,8,0:PLOT1,0,12:PLOT1,72,0:PLOT1,0,-24:PLOT1,-72,0:PLOT1,0,12:PLO
T0,72,0:PLOT1,8,0:RETURN
10070 PLOT0,-88,0:GOSUB10030:PLOT0,-88,0:RETURN
10110
PLOT1,0,-8:PLOT1,10,0:PLOT1,0,-72:PLOT1,-20,0:PLOT1,0,72:PLOT1,10,0:PL
OT0,0,-72:PLOT1,0,-8:RETURN
10140 PLOT0,0,88:GOSUB10110:PLOT0,0,88:RETURN
```




```
Low-Pass 2nd order filter with "Butterworth" reponse:

Cutoff Frequency  f = ------------------------
                      2 x SQRT(2 x pi x R x C)

Roll-off slope = 12dB per Octave, 40dB per Decade.
```

Figure 2. Comprehensive annotation can be produced.

a downward arrow followed by four presses of the '_' key makes a fair earth symbol. Lastly, as with all high quality products, batteries are included, just press 'B'.

The most important facility in any drafting program is the ability to erase mistakes as any honest draftsperson will readily testify. To carry out this function, first press 'E'. This will latch the current cursor position as the bottom left co-ordinate of a rectangle. Next move the cursor to the top right corner of the (imaginary) rectangle covering the area to be erased and press 'E' again and there, as they say, it is gone. In addition, the line drawing commands in shifted mode will erase any previously drawn lines and pressing the 'K' key twice will erase the entire screen.

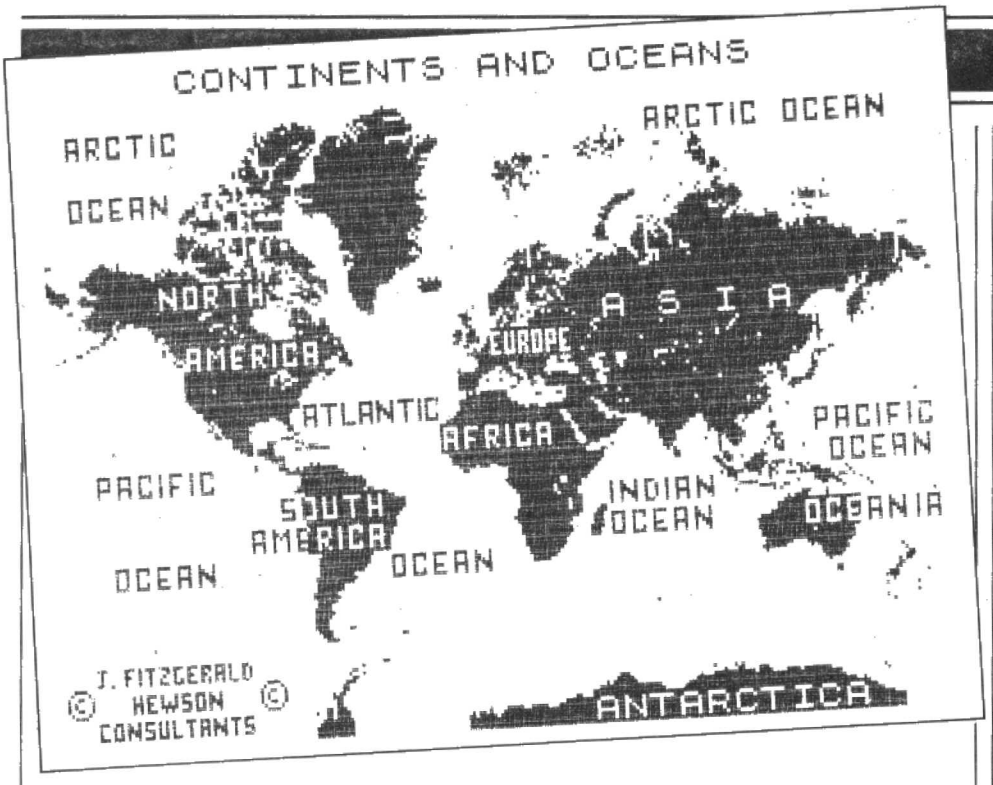
The command 'W' allows text to be written at the current cursor position. With a little practice, the position of text can be controlled very accurately and the labelling of components, the marking of IC pin numbers and the leaving of frightening copyright messages are all easily mastered. One word of warning, the delete key will remove anything in its path, treat it with care.

The 'S' and 'L' keys control the SAVeing and LOADing of the whole of the screen memory area. **Listing 1** contains three '*DIR' commands in lines 60, 9800, and 9905: these are intended for disk-based filing systems and cause circuit data and files to appear in the catalogue with the prefix 'C'. Inserting REM statements in these lines will make the program compatible with tone-based systems.

May we have more?

If your field of interest is in controlling RF circuits with your BBC computer, or perhaps multi-megawatt mains heaters via the Beeb the lack of predefined chokes and thyristors may well prove a cause for concern. Unfortunately, working in mode 0 does not leave a great deal of room for programs and even asking the computer to draw a humble circle is likely to provoke complaints about cramped work space. The components that form the basic set of available symbols coupled with the ability to draw lines and to position text will meet most needs so long as you do not work exclusively in CMOS or TTL. In addition it is possible to customise the program to any specific requirements; what is wrong with 'PROCEXCLUSIVEORGATE' , or 'PROCFULLADDER' for example.

The world is your oyster, is your imagination a strong enough pen knife to open it?



FROM PIXEL TO PRINT

Richard Sargent describes a routine that endows the Spectrum with a screen copy facility that is compatible with Epson printers.

Having gone to the expense of equipping your Spectrum with a parallel printer port and dot matrix printer it is still more than likely that you will be unable to produce a copy of the computer's screen display. As the humble ZX printer could at least manage some sort of screen dump this situation does not seem to be altogether satisfactory. Luckily if your printer is capable of operating in a bit image mode all that is required to produce a screen copy is a short piece of machine code software. The code was written for use with the Wafadrive interface as reviewed in our January issue, but is capable of operating with a wide range of printer interfaces.

The machine code routine will copy the screen (including the bottom two lines) to a dot matrix printer which has been selected to operate in the bit image mode or "dot addressable" graphics mode. On the Epson MX70/MX80 series it will produce an image in normal-density mode which is 108mm wide, but using the Epson RX80/FX80/100 series an additional image, 162mm wide, can be produced using the "CRT Graphics" mode. However, owners of early MX80s may be disappointed since that machine was first produced with a text-only ROM inside it. Epson clones should be capable of responding to the code provided, but in any event there are spaces left in the source listing so that alterations or insertions can be made to the printer-control instructions. It will, however, requre a substantial re-write to persuade the routine to work with matrix printers which have less than eight wires in their print head.

The program presented in **Listing 1** has been assembled for a 48K Spectrum on the grounds that few if any Wafadrive owners will have 16K machines. Screen-copy code is always quite lengthy because dot matrix printers require their data in a totally different format than that used in the Spectrum display file, and the bits have to be substantially re-arranged and collected in a holding area before they are in a fit state to be sent to the printer. The Spectrum COPY key cannot be used as the data which that sends out through the expansion port is quite useless to most

"Capable of operation with a wide variety of printer interfaces".

dot-matrix printers. The routine is 00302K bytes long and also uses the 256 bytes in the Spectrum's own printer buffer RAM as the temporary holding area. If you are unlikely to need the DOUBLE size copy (162mm wide) you needn't bother to enter the code from FEFCH onwards. It must also be conceded that the printer set-up code at the beginning of the listing is not as concise as it might be (too many CALL

LISTING 1

```
5B00            BUFFER    EQU 23296
4000            VDU       EQU 16384
4000
FDD0                      ORG 0FDD0H
FDD0                      ;LOAD 0FDD0H
FDD0
FDD0 CD93FE     S         CALL SING
FDD3 CD8FFE               CALL DUB
FDD6 00         SIZE      DB 0
FDD6
FDD7 CD64FE     EIGHT     CALL PRINTIT
FDDA 3E1B                 LD A,1BH
FDDC CD64FE               CALL PRINTIT
FDDF 3E41                 LD A,"A"
FDE1 CD64FE               CALL PRINTIT
FDE4 3E08                 LD A,8
FDE6 CD64FE               CALL PRINTIT
FDE9 CD59FE               CALL CR_LF
FDEC C9                   RET
FDED 00000000
FDF1 00                   DB 0,0,0,0,0
FDF2 00000000
FDF6 00                   DB 0,0,0,0,0
FDF6
FDF7 3E1B       NORMAL    LD A,1BH
FDF9 CD64FE               CALL PRINTIT
FDFC 3E32                 LD A,"2"
FDFE CD64FE               CALL PRINTIT
FE01 CD59FE               CALL CR_LF
FE04 C9                   RET
FE05 00000000
FE09 00                   DB 0,0,0,0,0
FE09
FE0A C5         SINGLE    PUSH BC
FE0B 3E1B                 LD A,1BH
FE0D CD64FE               CALL PRINTIT
FE10 3E4B                 LD A,"K"
FE12 CD64FE               CALL PRINTIT
FE15 0000                 DB 0,0
FE17 000000               DB 0,0,0
FE1A 3E00                 LD A,00H
FE1C CD64FE               CALL PRINTIT
FE1F 3E01                 LD A,01H
FE21 CD64FE               CALL PRINTIT
FE24 21005B               LD HL,BUFFER
FE27 0600                 LD B,0
FE29 7E         PLP       LD A,(HL)
FE2A CD64FE               CALL PRINTIT
FE2D 23                   INC HL
FE2E 10F9                 DJNZ PLP
FE30 CD59FE               CALL CR_LF
FE33 C1                   POP BC
FE34 C9                   RET
FE35 00000000
FE39 00                   DB 0,0,0,0,0
FE39
FE3A 3E1B       PARAMS    LD A,1BH
FE3C CD64FE               CALL PRINTIT
FE3F 3E2A                 LD A,"*"
FE41 CD64FE               CALL PRINTIT
FE44 3E04                 LD A,4
FE46 CD64FE               CALL PRINTIT
FE49 3E00                 LD A,0
FE4B CD64FE               CALL PRINTIT
FE4E 3E02                 LD A,2
FE50 CD64FE               CALL PRINTIT
FE53 C9                   RET
FE54 00000000
FE58 00                   DB 0,0,0,0,0
FE58
FE59 3E0D       CR_LF     LD A,0DH
FE5B CD64FE               CALL PRINTIT
FE5E 3E0A                 LD A,0AH
FE60 CD64FE               CALL PRINTIT
FE63 C9                   RET
FE63
FE64 C367FE     PRINTIT   JP PRINTER
FE64
FE67 C5         PRINTER   PUSH BC
FE68 F5                   PUSH AF
FE69 F5                   PUSH AF
FE6A 010200               LD BC,2
FE6D ED78       RDY       IN A,(C)
FE6F CB6F                 BIT 5,A
FE71 20FA                 JR NZ RDY
FE73 F1                   POP AF
FE74 47                   LD B,A
FE75 0E0E                 LD C,14
FE77 ED78                 IN A,(C)
FE79 0600                 LD B,0
FE7B 0E0A                 LD C,10
FE7D ED78                 IN A,(C)
FE7F C5                   PUSH BC
FE80 C1                   POP BC
FE81 0620                 LD B,20H
FE83 0E0A                 LD C,10
FE85 ED78                 IN A,(C)
FE87 F1                   POP AF
FE88 C1                   POP BC
FE89 C9                   RET
FE8A 00000000
FE8E 00                   DB 0,0,0,0,0
FE8E
FE8F 3E01       DUB       LD A,1
```

LISTING 1 – Continued

```
FE91 1801              JR DUB2
FE93 AF       SING     XOR A
FE94 32D6FD   DUB2     LD (SIZE),A
FE97 CDD7FD            CALL EIGHT
FE9A 0600              LD B,0
FE9C 0E00     COORD1   LD C,0
FE9E DD21005B          LD IX,BUFFER
FEA2 79       COORD2   LD A,C
FEA3 CB3F              SRL A
FEA5 6F                LD L,A
FEA6 78                LD A,B
FEA7 E630              AND 30H
FEA9 0F                RRCA
FEAA 67                LD H,A
FEAB 78                LD A,B
FEAC E60E              AND 0EH
FEAE 07                RLCA
FEAF 07                RLCA
FEB0 07                RLCA
FEB1 07                RLCA
FEB2 B5                OR L
FEB3 6F                LD L,A
FEB4 78                LD A,B
FEB5 E601              AND 1
FEB7 07                RLCA
FEB8 07                RLCA
FEB9 B4                OR H
FEBA F640              OR 40H
FEBC 67                LD H,A
FEBC
FEBD C5                PUSH BC
FEBE 0608              LD B,8
FEBE
FEC0 E5       ROTATE   PUSH HL
FEC1 1600              LD D,0
FEC3 3E08              LD A,8
FEC5 4E       F2       LD C,(HL)
FEC6 5B                LD E,B
FEC7 CB39     F3       SRL C
FEC9 1D                DEC E
FECA 20FB              JR NZ F3
FECC CB12              RL D
FECE 24                INC H
FECF 3D                DEC A
FED0 20F3              JR NZ F2
FED2 7A                LD A,D
FED3 DD7700            LD (IX+0),A
FED6 DD23              INC IX
FED8 E1                POP HL
FED9 10E5              DJNZ ROTATE
FEDB C1                POP BC
FEDC 0C                INC C
FEDD 0C                INC C
FEDE 79                LD A,C
FEDF FE40              CP 64
FEE1 38BF              JR C COORD2
FEE3 3AD6FD            LD A,(SIZE)
FEE6 B7                OR A
FEE7 2005              JR NZ F4B
FEE9 CD0AFE            CALL SINGLE
FEEC 1803              JR F5B
FEEE CDFCFE   F4B      CALL DOUBLE
FEF1 04       F5B      INC B
FEF2 04                INC B
FEF3 78                LD A,B
FEF4 FE30              CP 48
FEF6 38A4              JR C, COORD1
FEF8 CDF7FD            CALL NORMAL
FEFB C9                RET
FEFB
FEFB          ;-----------------------
FEFB
FEFC C5       DOUBLE   PUSH BC
FEFD CD3AFE            CALL PARAMS
FF00 CD37FF            CALL T_BUFF
FF03 CD59FE            CALL CR_LF
FF06 CD3AFE            CALL PARAMS
FF09 CD11FF            CALL B_BUFF
FF0C CD59FE            CALL CR_LF
FF0F C1                POP BC
FF10 C9                RET
FF10
FF11 0600     B_BUFF   LD B,0
FF13 21005B            LD HL,BUFFER
FF16 5E       B_LP8    LD E,(HL)
FF17 CB23              SLA E
FF19 CB23              SLA E
FF1B CB23              SLA E
FF1D CB23              SLA E
FF1F 0E04              LD C,4
FF21 CB23     B_LP9    SLA E
FF23 F5                PUSH AF
FF24 CB12              RL D
FF26 F1                POP AF
FF27 CB12              RL D
FF29 0D                DEC C
FF2A 20F5              JR NZ B_LP9
FF2C 7A                LD A,D
FF2D CD64FE            CALL PRINTIT
FF30 CD64FE            CALL PRINTIT
FF33 23                INC HL
FF34 10E0              DJNZ B_LP8
FF36 C9                RET
FF36
```

LISTING 1 – Continued

```
FF37 0600     T_BUFF   LD B,0
FF39 21005B            LD HL,BUFFER
FF3C 5E       LP8      LD E,(HL)
FF3D 0E04              LD C,4
FF3F CB23     LP9      SLA E
FF41 F5                PUSH AF
FF42 CB12              RL D
FF44 F1                POP AF
FF45 CB12              RL D
FF47 0D                DEC C
FF48 20F5              JR NZ LP9
FF4A 7A                LD A,D
FF4B CD64FE            CALL PRINTIT
FF4E CD64FE            CALL PRINTIT
FF51 23                INC HL
FF52 10E8              DJNZ LP8
FF54 C9                RET
FF54
FF55          FIN1     EQU $
0185          LEN1     EQU FIN1-S
```

PRINTITs), but at least it is clear, and if it is necessary to alter it to suit a non-Epson printer it is helpful to see what's going on.

The code from FE8FH onwards won't need changing, unless of course the entire program is relocated. To use the program with interfaces other than the Wafadrive, the PRINTIT vector will need changing, but more of that later.

Wafadriving

The Wafadrive doesn't need to be enabled in order to use the screen copy routine, it is therefore possible to operate by loading the routine from tape and work with a full-memory Spectrum. Assuming, though, that the Wafadrives are fully functional, CLEAR 64975H and load the code using a Hex-Loader. Save the program with SAVE *"a:dump",64976H,00389H then verify them by using VERIFY *"a:dump". Notice that the Wafadrive syntax doesn't require the word CODE to be used. The directory of the wafer should show that DUMP.BYT has been successfully filed, and can be loaded using LOAD *"a:dump".

The PRINTER routine at FE67H is worth looking at since it uses the abnormal instructions required to drive the Wafadrive electronics. As explained in the handbook, IN instructions are used even when data is to be sent OUT to the parallel port. The book gives examples of how to do this in BASIC, and here the instructions are translated into machine code. Thus

```
LD B,A
LD C,14
IN A,(C)
```

outputs the data byte, which is in the accumulator to port 14

```
LD B,0
LD C,10
IN A,(C)
```

sends a strobe to port 10

```
LD B,20H
LD C,10
IN A,(C)
```

cancels the strobe

As far as the software is concerned, this is as good a way of doing it as any other, as the eccentricity is in the hardware and that

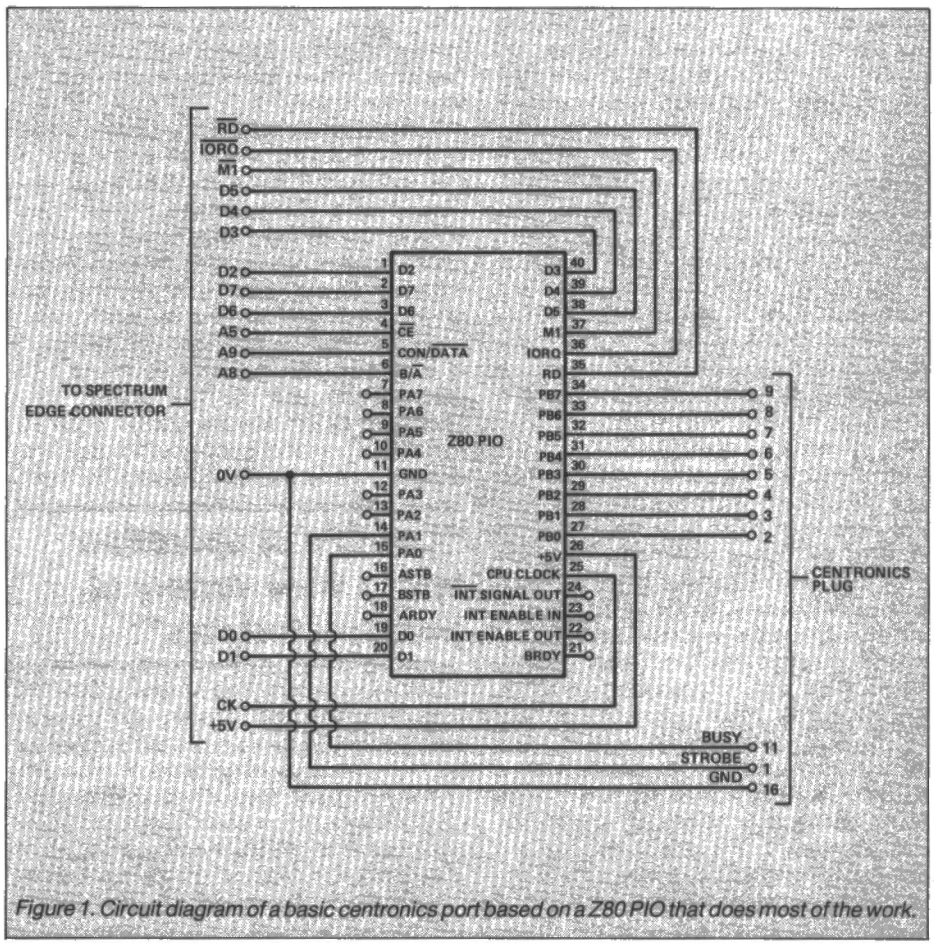
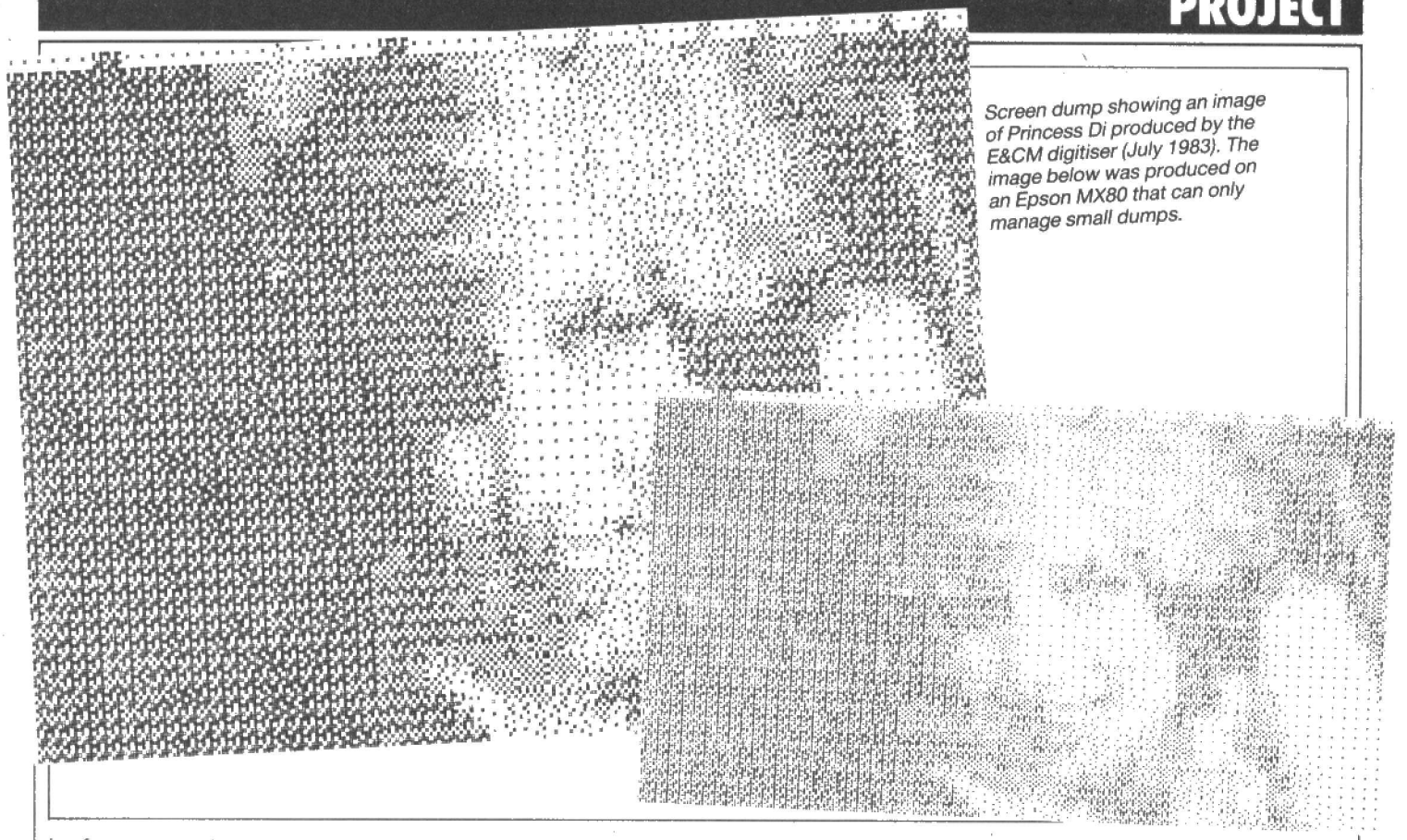


Figure 1. Circuit diagram of a basic centronics port based on a Z80 PIO that does most of the work.



Screen dump showing an image of Princess Di produced by the E&CM digitiser (July 1983). The image below was produced on an Epson MX80 that can only manage small dumps.

is of no concern to us.

Other interfaces

For other parallel ports forget about the PRINTER routine, and patch the address of your own printer routine into the PRINTIT vector at FE65H/FE66H. Most commercial Centronics interfaces will have a routine in their ROM which is the equivalent of PRINTER. Other general purpose peripherals containing either the Z80PIO chip or the 8255 PIA can be configured as Centronics ports.

The code for the Z80PIO is particularly simple. Assuming you use the "B" port as the data output, Bit1 of the "A" port as the strobe line and Bit0 of the "A" port as the connection to the printer BUSY line (**Figure 1**), then a single call to the code of **Listing 2** will configure the PIO correctly and **Listing 3** becomes the new PRINTER subroutine. The BC-register port addresses given assume that pin 4 is connected to the Spectrum's address line A5, pin 5 to A9 and pin 6 to A8, but the source listing can easily be changed to accommodate other arrangements.

Printers

Printing in high-resolution graphics mode requires a number of parameters to be passed to the printer, both prior to the printing session as a whole, and before the start of each individual print-line. These parameters are delivered at the appropriate moments by four subroutines called EIGHT, NORMAL, SINGLE and PARAMS. Between each routine some NOPs form padding which can be taken up if these routines need lengthening.

EIGHT is called once at the start of the screen dump. It sets the linefeed length to eight-dot vertical spacing, Epson code <ESC,41H,8>. A call to CR-LF also advances the paper and clears the printer's own line buffer.

NORMAL is called at the end of the screen copy, and sets the printer back to normal vertical spacing, <ESC,32H> on the Epson.

LISTING 2

```
FDB0                        ORG 0FDB0H
FDB0                        ;LOAD @FDB0H
FDB0
FDB0 01DFFE    SETPIO       LD BC,PACON
FDB3 3ECF                   LD A,0CFH
FDB5 ED79                   OUT (C),A
FDB7 3ED1                   LD A,0D1H
FDB9 ED79                   OUT (C),A
FDBB 01DFFF                 LD BC,PBCON
FDBE 3E0F                   LD A,0FH
FDC0 ED79                   OUT (C),A
FDC2 01DFFC                 LD BC,PADATA
FDC5 3E02                   LD A,2
FDC7 ED79                   OUT (C),A
FDC9 C9                     RET

FDCA 00000000
FDCE 0000                   DB 0,0,0,0,0,0
```

SINGLE is called 24 times because it is needed before each line. The parameter passed to the printer are <ESC,4BH> to set the normal bit image mode and <00,01> to indicate that 0100H bytes of image data will be sent to the printer. Sufficient NOPs are provided to allow changes to be made.

PARAMS is called instead of SINGLE if the double size screen dump is being produced. In this case the three bytes <ESC,2AH,4> set the printer to special graphics mode and <00,02> or 0200H bytes are sent to the printer for the longer line.

If your printer has been set to give an automatic linefeed after every carriage-return, then you should blank out the code at FE60H with three NOPs.

To copy a screen, execute the direct command RANDOMIZE USR 64976 (or RANDOMIZE USR 64979 for the double-size printout). Because of the way the Spectrum works the last two lines will be printed blank. If it is required to copy the entire screen, the routine must be called from within a program so that the bottom two lines are not wiped by the operating system.

The screen dumps shown on these pages show what can be achieved with the screen dump software. The increasing amount of applications software that produce graphic displays the software together with an appropriate interface and printer should find no shortage of applications.

LISTING 3

```
FEDF            PACON       EQU 0FEDFH
FFDF            PBCON       EQU 0FFDFH
FCDF            PADATA      EQU 0FCDFH
FDDF            PBDATA      EQU 0FDDFH
FDDF
FE67                        ORG 0FE67H
FE67                        ;LOAD @FE67H
FE67
FE67 C5         PIOPRINT    PUSH BC
FE68 F5                     PUSH AF
FE69 F5                     PUSH AF
FE6A CDB0FD                 CALL SETPIO
FE6D 01DFFC                 LD BC,PADATA
FE70 ED78       READY       IN A,(C)
FE72 CB47                   BIT 0,A
FE74 20FA                   JR NZ READY
FE76 F1                     POP AF
FE77 01DFFD                 LD BC,PBDATA
FE7A ED79                   OUT (C),A
FE7C 01DFFC                 LD BC,PADATA
FE7F AF                     XOR A
FE80 ED79                   OUT (C),A
FE82 C5                     PUSH BC
FE83 C1                     POP BC
FE84 3E02                   LD A,2
FE86 ED79                   OUT (C),A
FE88 F1                     POP AF
FE89 C1                     POP BC
FE8A C9                     RET
```

PIPE DREAMS

Mastering the use of QL pipes allows for efficient job to job communication releasing more of the computer's power. Adam Denning has the details.

An important aspect of QL multitasking which warrants discussion is job to job communication. Usually each job is entirely separate, like a series of independent programs running on a number of different machines, but often we will want to send information from one job to another. We may, for example, want to write a complicated printer driver with spooler, which could be written as one job to read from the file to be printed and buffer it, and another job to process all the printer codes and send the output to the printer. A program which is both more useful and more difficult would be one which allowed us to invoke a job from SuperBasic and pass a command string to it.

The standard method for job to job communication is the input/output device known as the pipe. This is opened for output at one end and subsequently opened for input at the other. The only problem with this is that each pipe is identified solely by its channel ID – there are no pipe filenames. This makes it extremely awkward as the only way of opening a pipe is to go through this process:

open pipe for output in main job
pass channel ID returned by IO_OPEN to subsidiary job
open pipe for input in subsidiary job

This means that we have to poke around in the machine's memory. We take advantage of the fact that, by finding a job's start address, we can get the values of all the registers for that job from a defined area in the job control block. We can find the subsidiary job's stack pointer, A7, and store the pipe's channel ID on the new job's stack. We then alter the stack pointer as appropriate and re-save it in the job's control area. This is a tedious process but there seems to be no other way.

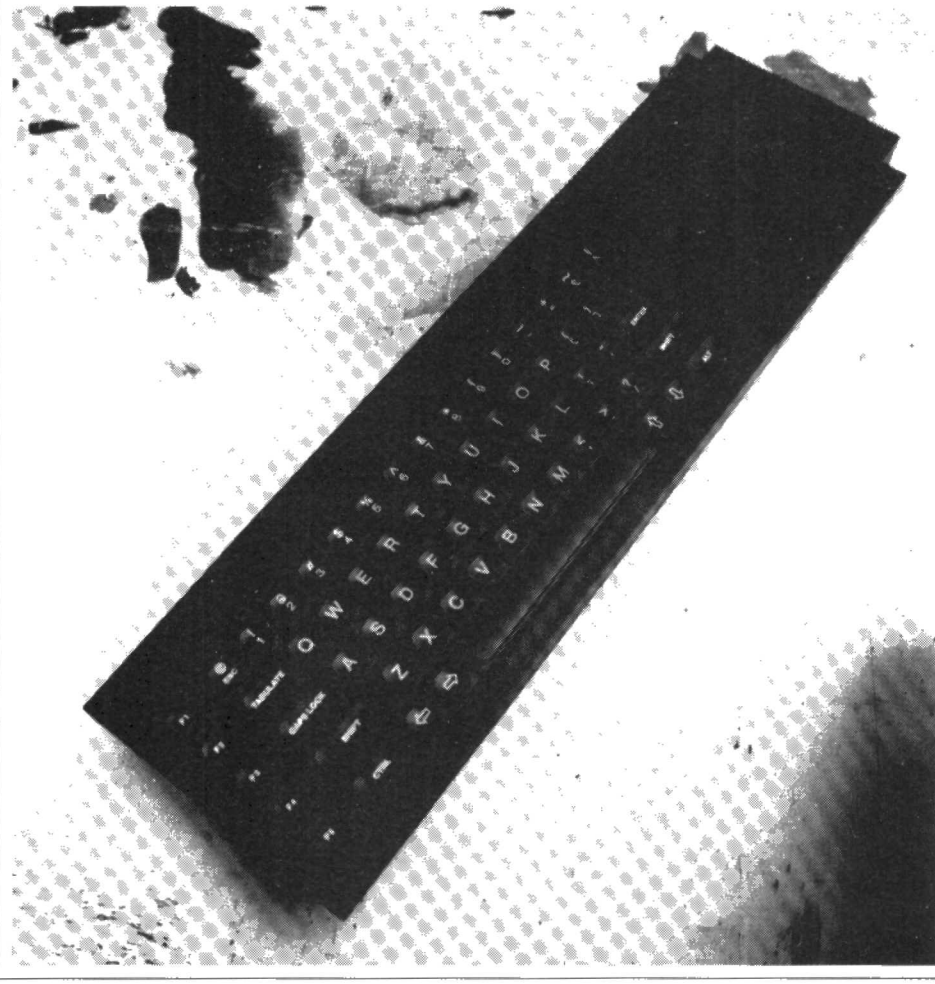
We'll demonstrate this technique first by writing a simple program consisting of two jobs. The first collects keyboard input and sends it down the pipe, while the second reads data from the pipe and prints it out to the screen.

The first program is shown in **Listing 1.** Firstly, this opens up a console device and saves its channel identity on the stack. It then opens a pipe device for output by specifying a pipe buffer in the device name ('PIPE_100') and using the OPEN_NEW key in the call to IO_OPEN. This ID is also saved on the stack and the subsidiary job file is opened for input. This file is called **mdv1_pipe_sub** here, but any valid filename will suffice. Once this file is opened its header is read into memory and its length and data space are extracted from it. These parameters are used in a call to MT_CJOB to create the new job – note that we clear A1 with MOVE.L #0,A1 here, when in a real program we would have used the shorter SUBA.L A1,A1 instruction instead. The new job's ID and starting address are saved on the stack and the file is loaded into its allocated area before the file is closed.

We have previously opened the pipe device for output, but before we can pass it to the subsidiary job we must open it for input. This is done by altering the device name to 'PIPE' as opposed to 'PIPE_buflen' and calling IO_OPEN with the ID for the job for which the channel is being opened in D1 and the pipe output device channel ID in D3. We change the device name by altering the word containing its length. This fools QDOS into thinking that the name is only four characters long.

"We take advantage of the fact that, by finding a job's start address, we can get the values of all the registers from a defined area in the job control block".

Now comes the interesting bit. We take



LISTING 1.

```
                        * A program to demonstrate job to job communication
                        * The main job
                        
                        * By Adam Denning (C) 1984 Adam Denning
                        
00000001B   ESCAPE      EQU     27              Ascii Escape code
0000000F    BUF_LEN     EQU     15              length of buffer
                        
00000068    JOB_AREA    EQU     $68             job control area length
0000005C    SAVE_USP    EQU     $5C             offset from JCB of saved A7
                        
00000040                DATA    64              enough for stack and buffer
                        INCLUDE 'adv1_header_asm'
                        
0 00000000 601E         BRA.S   J_START          Ignore standard format code
0 00000002 00000000     DC.L    0
0 00000006 4AFB         DC.W    $4AFB
0 00000008 0009         DC.W    9
0 0000000A 504950455F40 DC.B    'PIPE_MAIN',0
                        
0 00000014 0000 PBLOCK  DC.W    0               No border
0 00000016 0400         DC.W    $0400           Green paper black ink
0 00000018 01B8         DC.W    440             Window width
0 0000001A 000A         DC.W    10              Window height
0 0000001C 0024         DC.W    36              Window X origin
0 0000001E 000F         DC.W    15              Window Y origin
                        
0 00000020 43FAFFF2 J_START LEA.L   PBLOCK,A1   Open up console device
0 00000024 347800C6     MOVE.W  UT_CON,A2
0 00000028 4E92         JSR     (A2)
0 0000002A 2F08         MOVE.L  A0,-(A7)        Save console ID on stack
                        
0 0000002C 41FA00CA     LEA.L   PIPENAME,A0     Open output pipe
0 00000030 72FF         MOVEQ   #-1,D1          for this job
0 00000032 7602         MOVEQ   #OPEN_NEW,D3    with exclusive output rights
0 00000034 7001         MOVEQ   #IO_OPEN,D0
0 00000036 4E42         TRAP    #2
0 00000038 2F08         MOVE.L  A0,-(A7)        Save PIPE ID on stack
                        
0 0000003A 41FA00AC     LEA.L   JOB_FILE,A0     Open subsidiary job file
0 0000003E 72FF         MOVEQ   #-1,D1          for this job
0 00000040 7601         MOVEQ   #OPEN_INS,D3    with shared input rights
0 00000042 7001         MOVEQ   #IO_OPEN,D0
0 00000044 4E42         TRAP    #2
0 00000046 2F08         MOVE.L  A0,-(A7)        Save file ID on stack
                        
0 0000004B 7047         MOVEQ   #FS_HEADR,D0    Read the file's header into RAM
0 0000004A 740F         MOVEQ   #BUF_LEN,D2
0 0000004C 43FA00B4     LEA.L   BUFFER,A1       starting at BUFFER
0 00000050 76FF         MOVEQ   #-1,D3          with infinite timeout
0 00000052 4E43         TRAP    #3
                        
0 00000054 41FA00AC     LEA.L   BUFFER,A0
0 00000058 72FF         MOVEQ   #-1,D1          Create subsidiary job to this one
0 0000005A 2410         MOVE.L  (A0),D2         code length in D2
0 0000005C 26280006     MOVE.L  6(A0),D3        data length in D3
0 00000060 227C00000000 MOVE.L  #0,A1           default starting address
0 00000066 7001         MOVEQ   #MT_CJOB,D0
0 00000068 4E41         TRAP    #1
                        
0 0000006A 2F01         MOVE.L  D1,-(A7)        Save new job ID on stack
0 0000006C 2F08         MOVE.L  A0,-(A7)        Save its base address on stack
                        
0 0000006E 2248         MOVE.L  A0,A1
0 00000070 7048         MOVEQ   #FS_LOAD,D0     Load job file into its area
0 00000072 206F0008     MOVE.L  8(A7),A0        Get job file's channel ID
0 00000076 76FF         MOVEQ   #-1,D3          infinite timeout
0 00000078 4E43         TRAP    #3
                        
0 0000007A 7002         MOVEQ   #IO_CLOSE,D0
0 0000007C 4E42         TRAP    #2              Close job file
                        
0 0000007E 262F000C     MOVE.L  12(A7),D3       Get output pipe channel ID
0 00000082 222F0004     MOVE.L  4(A7),D1        Get new job ID
0 00000086 41FA0070     LEA.L   PIPENAME,A0     open input pipe for new job
0 0000008A 5950         SUBQ.W  #4,(A0)         Alter to input pipe name
0 0000008C 7001         MOVEQ   #IO_OPEN,D0
0 0000008E 4E42         TRAP    #2
                        
0 00000090 225F         MOVE.L  (A7)+,A1        Get new job's base address
0 00000092 93FC00000068 SUBA.L  #JOB_AREA,A1    Point to start of job control area
0 00000098 D3FC0000005C ADDA.L  #SAVE_USP,A1    Point to saved USP
                        
0 0000009E 2451         MOVE.L  (A1),A2         Get new job's USP in A2
0 000000A0 2508         MOVE.L  A0,-(A2)        Save input pipe ID on new
0 000000A2 228A         MOVE.L  A2,(A1)         job's stack
                                                save new job's new USP value
0 000000A4 700A         MOVEQ   #MT_ACTIV,D0
0 000000A6 221F         MOVE.L  (A7)+,D1        Activate subsidiary job
0 000000A8 7401         MOVEQ   #1,D2
0 000000AA 7600         MOVEQ   #0,D3           with a priority of 1
0 000000AC 4E41         TRAP    #1              and a timeout of zero
                        
0 000000AE 221F         MOVE.L  (A7)+,D1
                                                remove file ID from stack
0 000000B0 206F0004 SEND_KEYS MOVE.L 4(A7),A0
0 000000B4 7001         MOVEQ   #IO_FBYTE,D0    Get console ID
0 000000B6 76FF         MOVEQ   #-1,D3          read a byte from it
0 000000B8 4E43         TRAP    #3              with infinite timeout
                        
0 000000BA 0C01001B     CMPI.B  #ESCAPE,D1
0 000000BE 6716         BEQ.S   JOB_END         Is it 'ESC'?
                                                leave if so
0 000000C0 3F01         MOVE.W  D1,-(A7)
0 000000C2 7005         MOVEQ   #IO_SBYTE,D0    Save byte
0 000000C4 206F0002     MOVE.L  2(A7),A0        and send it to pipe
0 000000C8 4E43         TRAP    #3              Get pipe ID
                        
0 000000CA 321F         MOVE.W  (A7)+,D1
0 000000CC 7005         MOVEQ   #IO_SBYTE,D0    Retrieve byte
0 000000CE 206F0004     MOVE.L  4(A7),A0        and send it to console
0 000000D2 4E43         TRAP    #3              get console channel ID
                        
0 000000D4 60DA         BRA.S   SEND_KEYS
                                                repeat
0 000000D6 7002 JOB_END MOVEQ   #IO_CLOSE,D0
0 000000D8 4E42         TRAP    #2              Close console
                        
0 000000DA 205F         MOVE.L  (A7)+,A0
0 000000DC 7002         MOVEQ   #IO_CLOSE,D0    Close output pipe
0 000000DE 4E42         TRAP    #2
                        
0 000000E0 7005         MOVEQ   #MT_FRJOB,D0
0 000000E2 72FF         MOVEQ   #-1,D1          Kill
0 000000E4 7600         MOVEQ   #0,D3           this job and all its subsidiaries
0 000000E6 4E41         TRAP    #1              with no returned errors
                        
0 000000E8 000D JOB_FILE DC.W   13
0 000000EA 6D6476315F70 DC.B    'mdv1_pipe_sub',0
                                                ...or whatever you choose
0 000000F8 0008 PIPENAME DC.W   8
0 000000FA 504950455F31 DC.B    'PIPE_100'
                        
00000102    BUFFER      EQU     *
                        
                        END

****** TOTAL ERRORS    0 (line    0)
**** TOTAL WARNINGS    0 (line    0)
      memory usage   12 kbytes
```

LISTING 2.

```
                        * A program to demonstrate job to job communication
                        * The subsidiary program
                        
                        * By Adam Denning (C) 1984 Adam Denning
                        
                        INCLUDE 'mdv1_header_asm'
00000020                DATA    32              enough for stack
                        
0 00000000 601C         BRA.S   J_START
0 00000002 00000000     DC.L    0
0 00000006 4AFB         DC.W    $4AFB
0 00000008 0008         DC.W    8
0 0000000A 504950455F53 DC.B    'PIPE_SUB'
                        
0 00000012 0000 PBLOCK  DC.W    0               No border
0 00000014 0700         DC.W    $0700           White paper black ink
0 00000016 01B8         DC.W    440             Window width
0 00000018 0096         DC.W    150             Window height
0 0000001A 0024         DC.W    36              Window X origin
0 0000001C 0032         DC.W    50              Window Y origin
                        
0 0000001E 43FAFFF2 J_START LEA.L   PBLOCK,A1   Open up a screen device
0 00000022 347800C8     MOVE.W  UT_SCR,A2
0 00000026 4E92         JSR     (A2)
0 00000028 2F08         MOVE.L  A0,-(A7)        Save screen channel ID
                        
0 0000002A 206F0004 READ_KEY MOVE.L 4(A7),A0    Get input pipe ID
0 0000002E 7001         MOVEQ   #IO_FBYTE,D0    Read a byte from it
0 00000030 76FF         MOVEQ   #-1,D3
0 00000032 4E43         TRAP    #3
                        
0 00000034 2057         MOVE.L  (A7),A0         Get screen channel ID
0 00000036 7005         MOVEQ   #IO_SBYTE,D0    send a byte to it
0 00000038 4E43         TRAP    #3
                        
0 0000003A 60EE         BRA.S   READ_KEY
                        
                        END

****** TOTAL ERRORS    0 (line    0)
**** TOTAL WARNINGS    0 (line    0)
      memory usage   12 kbytes
```

the job's base address off the stack and find the saved value of A7 for that job by subtracting the length of the job control area and adding the required offset. This could have been done more efficiently by

```
SUBA.L  #JOB_AREA-SAVE_USP,A1
```

rather than

```
SUBA.L  #JOB_AREA,A1
ADDA.L  #SAVE_USP,A1
```

but it wouldn't be so clear. Using the address now in A1 we load A2 with the subsidiary job's saved user stack pointer (A7) and temporarily use it as a stack pointer through which to save the pipe device's channel ID. We then put the new value of A2 back into the subsidiary job's saved USP area and activate the subsidiary job.

The last section enters a loop in which a character is read from the console and sent to the pipe. To show that everything is working properly we also send the character to the console, so that each keypress is echoed by both jobs. When the key pressed was ESCape, the channels are closed and the jobs are killed.

Jobbing printer

The subsidiary job is a lot smaller as **Listing 2** reveals. A screen device is opened and its channel ID is saved on the stack. We know that immediately above this is the pipe channel ID passed to this job by its owner, so we retrieve that and enter a loop in which every character read from the input pipe is transferred to the screen. This job never ends as it is killed by its own when ESCape is pressed. Note that as we have not included any traps to activate the cursor in the main job's console device, this program needs to be activated using EXEC_W.

"... as we have not included any traps to activate the cursor in the main job's console device this program needs activating by EXEC_W".

So the cardinal rules for opening a pipe device are as follows:

1 Open for output first, using full device name (PIPE_buflen) for the main job. Keep the channel ID returned by IO_OPEN safe. The key in D3 for IO_OPEN should be 'open as new exclusive file'.

2 Open pipe for input by truncating the name to 'PIPE', putting the job ID for which the input pipe is to be opened in D1 and the output pipe channel ID in D3, and then calling IO_OPEN.

3 Save the channel ID returned by this on the subsidiary job's stack.

One problem is that if the main job kills itself, as in this case, there is a chance that the subsidiary job which is processing the information may be killed before it has read all the information from the pipe. Probably the best thing to do here is to close the output pipe and suspend the main job. When a read from the input pipe in the subsidiary job returns 'end of file' the main job can be released, where it can kill itself and the subsidiary job.

If you want to open a pipe from BASIC, then use OPEN_NEW#chan, PIPE_buflen. The channel ID of this output pipe will be stored at (chan * $28 + $30(A6)(A6)) **when in BASIC,** but it can be found from another job by finding the base address of BASIC from the GV_BASIC system variable ($28010) and using this in place of A6. This channel ID can then be used in an IO_OPEN call to open the pipe for input.

Two very useful things to have on the QL would be a routine which 'switches the printer on' so that all output to SuperBASIC channel #1 is also sent to the printer and a routine which allows you to activate a job from SuperBASIC with a command string and possibly some channel IDs passed to it. We are working on these ideas and hope to be able to have them in this magazine early this year.

BBC NETWORKING

A low cost alternative to Econet

Paul Beverley shows how the BBC micro's inbuilt RS423 serial port can form the basis of a low cost network system that offers a wide range of facilities.

This series of articles will examine the serial communication system known as the RS423 and the principles of serial data transfer and then look at a range of possible applications. Firstly, the basic ideas will be explained and then a specific application for using the serial port to provide a low cost networking facility for schools or other users who have a number of computers in one area. For the cost of a few metres of cable and few DIN plugs, it is possible to provide a facility for sending programs from master computer, probably with disks, to a number of other computers.

Sending programs back to the master computer is not so easy, but this will be tackled in detail next month. Then, in succeeding articles, various other applications will be presented including ideas for using the BBC computer as a colour graphics terminal to a mini or mainframe computer and for using one BBC machine as a second processor to another! There are lots of things that can be done using the RS423 port to connect up to a friend's computer and you can send messages to each other and even play computer games against one another. To take this idea further, the computers may be linked via an acoustic coupler and exchanges made down the telephone line. These might include latest programs or some text which has been prepared on a wordprocessor. This can be done without using a modem, which might cost up to £60, but by simply building an acoustic coupler which can be made for less than £5.

Serial Data Transfer principles

"Data transfer" – By this we mean that one computer wants to send information to one or more computers. This information usually takes the form of 8-bit bytes, and could represent programs, text in the form of ASCII characters, or perhaps numerical data.

"Serial" – Within the computer itself, data is moved around on eight parallel data lines. When we want to send information from one computer to another, it is not always easy or convenient to have eight cables to each of a number of computers. Serial communication then means that we take each 8-bit byte and send the bits, one after the other, down a single communications channel. Then at the other end these eight bits are put back into a single byte ready to be transferred to the appropriate place in the computer.

There is a potential problem in getting the receiving computer to work out from a continuous stream of bits where one byte ends and the next begins. There are two basic ways of doing this which are referred to as synchronous and asynchronous. In the first instance a separate clock signal is sent which carries the information as to which bit is which. This, of course, requires an extra communication channel to carry the synchronising signal, which would make it virtually impossible to use over an ordinary telephone network.

Alternatively, asynchronous data transfer uses just one communications channel and informs the receiving computer when a new byte is about to start by sending a single bit, referred to as the start bit, immediately prior to the data byte. After sending the eight bits, it sends a stop bit and delays a short space of time so that the receiving computer will realise that this is the end of a byte and will be ready to receive the next start bit. This means that to send eight bits of data will take a time equivalent to approximately eleven bits (8 data bits, 1 start, 1 stop and approximately 1 delay bit).

The BBC's Econet system uses the synchronous technique for data communication. All the computers in the network are linked to a common clock source, and have a common data line on which any computer can 'speak' to any other. The RS423, however, is asynchronous and uses separate send and receive lines for the data. This is because it is electronically more complex to allow different computers to send data on the same line. Each computer would have to sense whether there was a signal on the line before going into the send mode. There is also the possibility that two computers might try to come onto the line at exactly the same time. The intention of the RS423 system is that communication will be on a one to one basis; two computers, or a computer and a terminal, exchanging data with one another. However, as will be shown later, it is also possible to use the RS423 system to broadcast data to a number of receiving stations.

The BBC microcomputer has a third serial communication system: the cassette interface. Indeed, the cassette system uses some of the same hardware as the RS423. The block diagram of both these systems is shown in **Figure 1.** At the heart of these circuits is a 6854 ACIA. (Asynchronous Communications Interface Adaptor). The job of the ACIA is to take in the parallel data and transfer it into a serial bit stream, adding the start and stop bits automatically. In the other direction, it takes in the serial stream of bits and works out which are the eight data bits which the processor then transfers on the internal data bus. The serial ULA is a chip which, amongst other things, switches the serial data streams between the RS423 and the cassette interface. It also takes in a clock pulse which determines the speed at which the ACIA works and, if necessary, divides the frequency down before passing it on to the ACIA.

"There are lots of things that can be done with the RS423 port including building an acoustic modem for less than five pounds".

Electrically the RS423 uses different voltage levels from those used by the internal data lines. Therefore buffers are required to convert the voltage levels to and from the RS423 interface. The cassette interface has its own input/output hardware, but we are not concerned with that in this present article.

From the software point of view, to send data out on the RS423 simply use the normal output facilities, PRINT or VDU (or

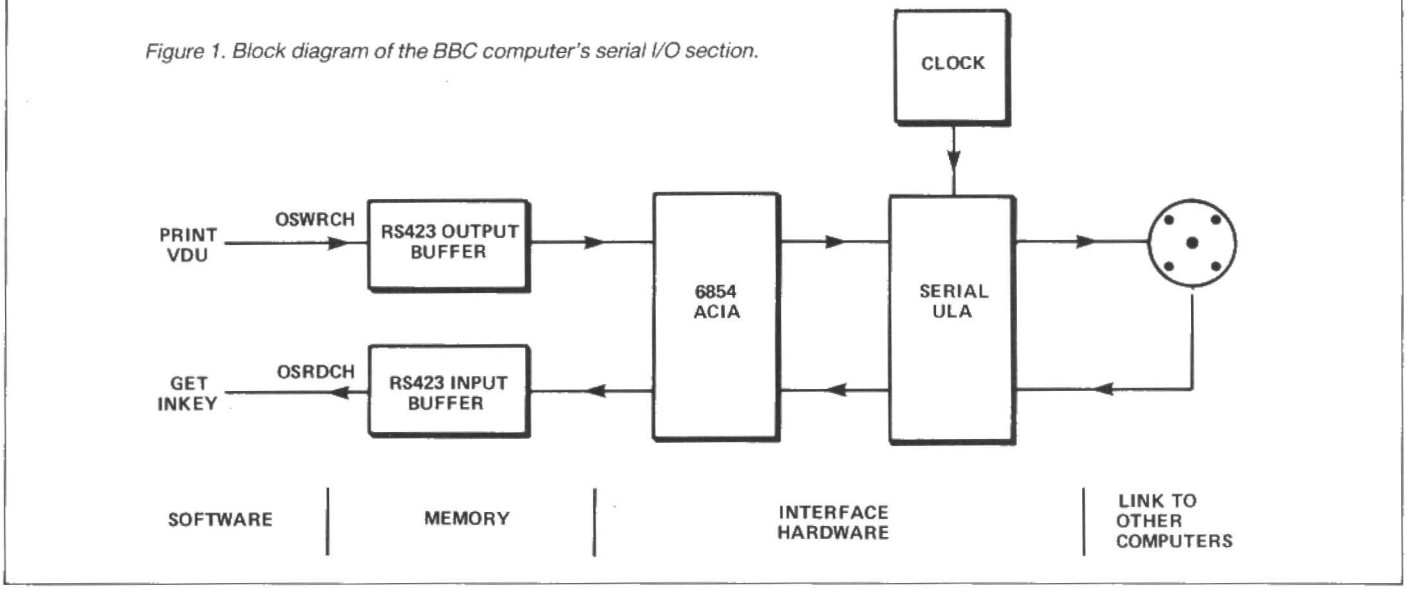


Figure 1. Block diagram of the BBC computer's serial I/O section.

OSWRCH or OSASCII in assembly language) and the operating system takes care of feeding these characters to the ACIA. This is done automatically under interrupt so that the computer does not have to sit and wait until each character is accepted by the ACIA. Instead they are fed into an area of memory referred to as a buffer. The operating system takes them from the buffer and transfers them to the ACIA as and when it is ready. This buffer can take up to 192 characters before it is full. When it is full, the operating system halts the program which is generating the characters. The fact that the program has stalled is indicated to the user by switching on both the shift-lock and the caps-lock lights.

When serial data comes into the ACIA it is stored in a 256 byte buffer until it is accessed by the program. This assumes that this function has been enabled by using the appropriate FX call. The information in the buffer can be accessed by using GET or INKEY (or OSRDCH or OSBYTE 129 in assembly language). If this input buffer becomes full, the only way to avoid losing characters is to stop the sending computer from sending any more. This can be achieved by using what are referred to as "handshake lines" on the RS423 interface. The ACIA on the sending computer can be halted by sending a signal in on the CTS line (clear to send). Therefore when the input buffer on the receiving computer becomes full it sends out a signal on the RTS line (ready to send) which is connected to the CTS line on the sending computer and thus halts the ACIA.

"There is no need to use special (expensive) co-axial cable – the cheapest flex will do".

The disadvantage of using handshaking is that it means using an extra line along with the data line, and one is needed in each direction making a total of four lines plus an earth line. To avoid the use of handshake lines then data has to be sent slowly enough so that the receiving computer's input buffer never becomes full. If this did happen then characters would be lost. If, however, the CTS input line on the sending computer is left unconnected then it assumes a logic state which indicates that it is *not* clear to send, causing it to stall and not send any characters at all. The trick therefore is to take the RTS output and feed it back into its own CTS input line. This convinces the sending computer that it is always clear to send.

FX/OSBYTE Calls

There are in fact 13 different FX or OSBYTE calls associated with the serial system and full details of these are given in the Advanced User Guide, page 309. For our purposes though we will only require the first few and, should the need for more complex calls arise, they will be explained as and when needed. The five most commonly used are shown in **Table 1.**

RS423 Network

There is very little hardware involved in implementing a network to transfer data from a master computer to a number of other computers. All that is required is to connect the data output line of the master computer to each of the data input lines on the other computers – all in parallel. This is shown in **Figure 2.** The only other connection to make is the RTS and CTS lines which have to be linked together on the master computer.

One potential problem is that on the input of each computer there are two protection resistors. These are included so that if an RS232 device is linked to the BBC computer, the higher voltages which it produces (+12V and –12V) will not damage the buffer chip. When a number of computers are put in parallel then these resis-

TABLE 1.

*FX2 – This call selects which is the input channel to be used by GET or INKEY. *FX 2,1 selects the RS423 input buffer and *FX 2,2 selects the keyboard input buffer. If this latter call is made then any characters arriving on the RS423 will automatically be put into the RS423 input buffer until they are required. The default value of this call, which is the value assumed when the computer is first switched on, or after a break, is *FX 2,0. In this mode the input comes from the keyboard, but the RS423 input buffer is disabled. This means that the computer ignores all input that appears at the ACIA.

*FX 3 – This call selects which channels are used as output when PRINT or VDU are used. Unlike input, where only one channel can be used at a time, output can be sent to any or all of the output channels at the same time. There are three basic channels available – the screen, the printer, and the RS423. The situation is confused somewhat by the fact that the serial output can be used as a printer port, instead of using the parallel printer port. In other words the printer output is selected, but the data is rerouted to the RS423. Since, however, a serial printer is not required, these complications will be ignored. The only calls needed therefore are *FX 3,0 which selects the screen only, *FX 3,1 (or *FX 3,5) which enables both the screen and the RS423, and *FX 3,7 which enables the RS423 only.

*FX 5 – This can be used to switch the printer type from parallel to serial, but this will not concern us for data communications purposes.

*FX 7 – This selects the receive baud rate, ie it sets the number of bits per second expected by the ACIA. It can be between 75 baud (*FX 7,1) and 19,200 baud (*FX 7,8).

*FX 8 – This is the equivalent of *FX 7, but for transmission purposes. Thus, for example, *FX 8,1 will set the output transmission rate at 75 baud.

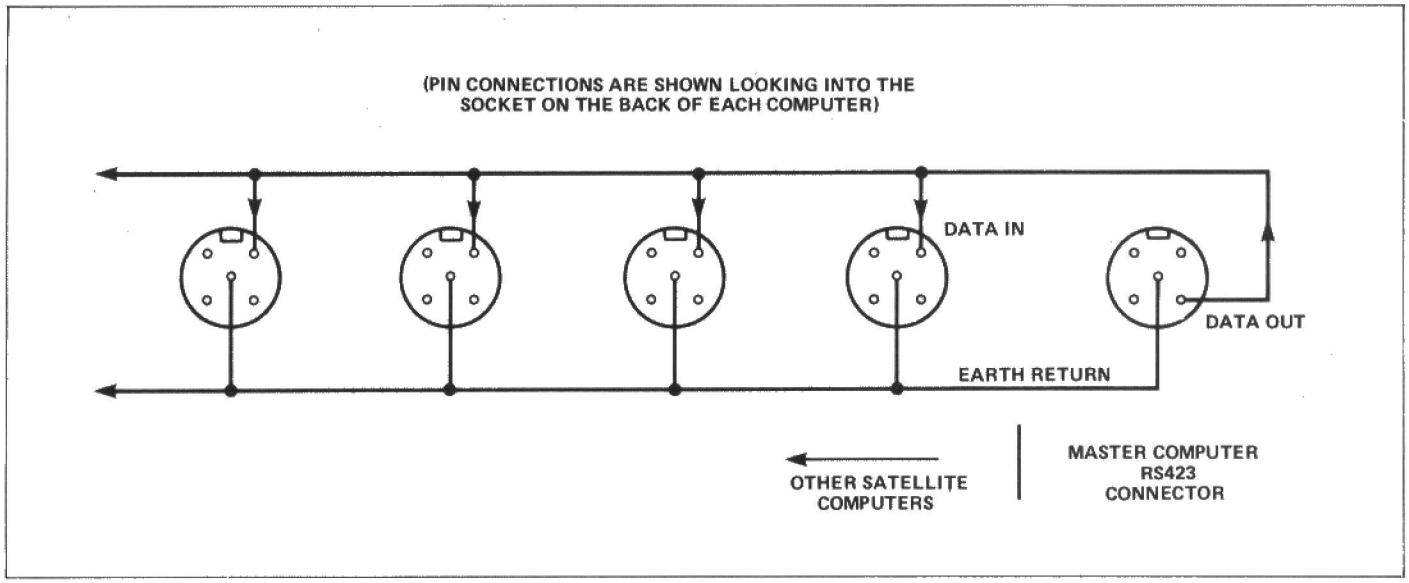


Figure 2. An RS423 network can be formed by connecting the micro's I/O lines in parallel.

tors will form a significant load on the RS423 output of the master computer. If the number of receiving computers exceeds say six, then it is advisable to disconnect the 3k3 resistors on some, if not all, of the receiving computers as shown in **Figure 3(a).** This can be done quite easily (and reversibly) with a pair of side-cutters. The resistor in question (R93) is the left-hand of the three resistors immediately behind the RS423 connector, above the 88LS120 interface chip (**Figure 3(b)**). All you need to do is to cut the wire at the top of this resistor. This can be done in such a way that if you wanted to re-connect it, the wires which had been cut could be moved into contact and a blob of solder added. This is really not as painful or dangerous as it sounds!

The cable used for carrying the data can be almost anything. The cheapest is probably twin-core speaker flex. Provided these cables are not routed near to other communications cables, such as telephone lines, there is no need to use coaxial cable. Indeed, using coaxial cable produces a capacitive loading on the sending device, especially significant at higher speeds.

In order to use this system, having wired it up, all you need is to use a few of the FX calls. The easiest way of doing it is to program the function keys on the master computer as shown in **Table 2.** Assuming an appropriate data transfer speed is selected, all that has to be done is to get the users of those computers that are to receive the program to type in *FX 2,1 and press return. Then on the master computer, load in the program to be sent and press f0.

The effect of the string shown as *KEY 0 is to select both the screen and the RS423 as output, and then print "NEW". This word is printed on the screen of the master computer, but on the receiving computers it is acted upon since it is as though it had been typed in from the keyboard. This clears any old program from the computer and makes it ready to receive the new one. The program is then listed both to the master screen and to the receiving computers. To them it is just as if someone were typing it in on the keyboard but doing it extremely

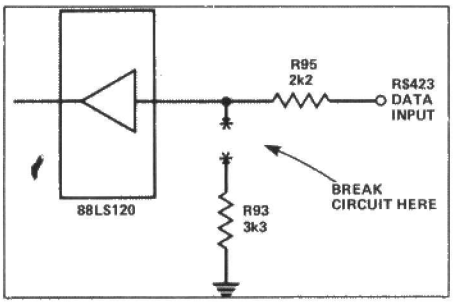


Figure 3(a). Modification to RS423 input circuitry.

quickly! This is then followed by a clear screen command (not strictly necessary) and the instruction *FX 2,0 which sets the receiving computers back to the condition where their keyboards can be used. The final call switches off the RS423 output.

Because commands can be sent down

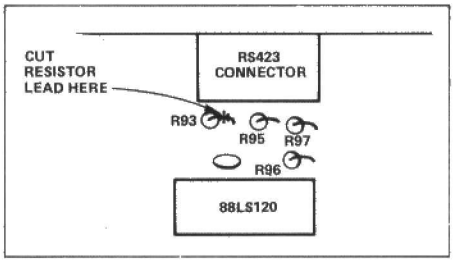


Figure 3(b). Location of circuit break.

to the receiving computers it is possible, as shown in the *KEY 1 line in **Table 2,** to send down a program and also cause it to start running straightaway. This is done by the PRINT "RUN" command. The only thing to realise here is that the keyboards on the receiving computers are still disabled. Therefore it is necessary to put a *FX 2,0 into the program itself before the users can use the keyboards for input.

It must be remembered that before any program is sent the correct speed must be selected for both transmitting and receiving. The speed at which they are sent must be slow enough so that the receiving computers do not miss any characters. The default value of 9600 baud is far too fast without using handshake lines, and there is no simple way to implement a multiple handshaking facility. Therefore you simply have to choose a low enough speed – 1200 baud is usually adequate, so you type in *FX 7,4 on the receiving computers and *FX 8,4 on the master computer. The receiving computers will also respond faster to the incoming characters if they are switched into the teletext mode. This can be done by adding PRINT "MODE 7" into the key definition. It may be possible to run at 2400 baud but this will have to be discovered by trial and error.

Finally you have to beware of programs that have been compressed, producing some very long multi-statement lines. When these lines are listed out in full, some of them may be longer than the 256 bytes that the BASIC input buffer can accept. If such a line is sent, the receiving computers will all beep at you and you will simply have to split the line before sending the program again.

Next month we will see how to implement the other half of this network system – to send programs and data from the satellite computers back to the master. This is not quite as simple, but we will offer two possible solutions. We will then go on to look at a message sending program – a great deal of fun can be had by sending messages from one room to another or maybe even from one house to another over the back garden wall! You might even find some "useful" applications for it.

TABLE 2. Possible key definitions for broadcasting programs.

```
*KEY 0  *FX3,5:M P. "NEW":M L.:M P. CHR$12;"*FX2,0":M *FX3,0:M
*KEY 1  *FX3,5:M P. "NEW":M L.:M P. "RUN":M *FX3,0:M
*KEY 3  *FX8,4:M
```

How to avoid the misery of SLIPPED DISKS

This utility builds up a super catalogue of all the programs on a collection of disks and allows them to be listed either alphabetically or by directory type.

Program by Mike Williams

Have you ever wanted to find a program, only to discover that you couldn't remember which disk it was on? Of course it was even more difficult with cassettes, but that's not much comfort as you search from disk to disk, wondering if somehow the elusive gem has been deleted. 'I must get my disks catalogued' you mutter to yourself. Here is the program to do just that; it makes life easier and relieves much of the worry, not to say boredom, caused by slipped disks.

When this utility is run, all that has to be done is to feed in your disks one after the other. Once this process is finished a list of all programs is shown on the screen, grouped either by disk or alphabetically. The list can be saved onto disk as a file, or printed out as hard copy. The program allows a choice of which directories to catalogue, so that if all utilities are in, say, directory U, then it is easy to catalogue them and to keep a separate filed list.

Now for the program itself. It is given in **Listing 1.** The program gives full on screen instructions and there is no need to delve into the mechanics of it. On the other hand you might well wish to customise the program to suit your own requirements, and for that reason a description of the various procedures follows.

Procsetup

This does the usual preliminary setting up of variables, including those involved in getting the catalogue off disk and writing it into memory (lines 360-390).

Line 400 dimensions the arrays which are going to hold the program and disk names. These are set to take up to 800 programs and 50 disks.

PROCload data

This asks if you want to load a catalogue file which has been previously created. Once loaded, the catalogue can be viewed, added to or printed.

PROCcatalogue

This procedure calls three others:

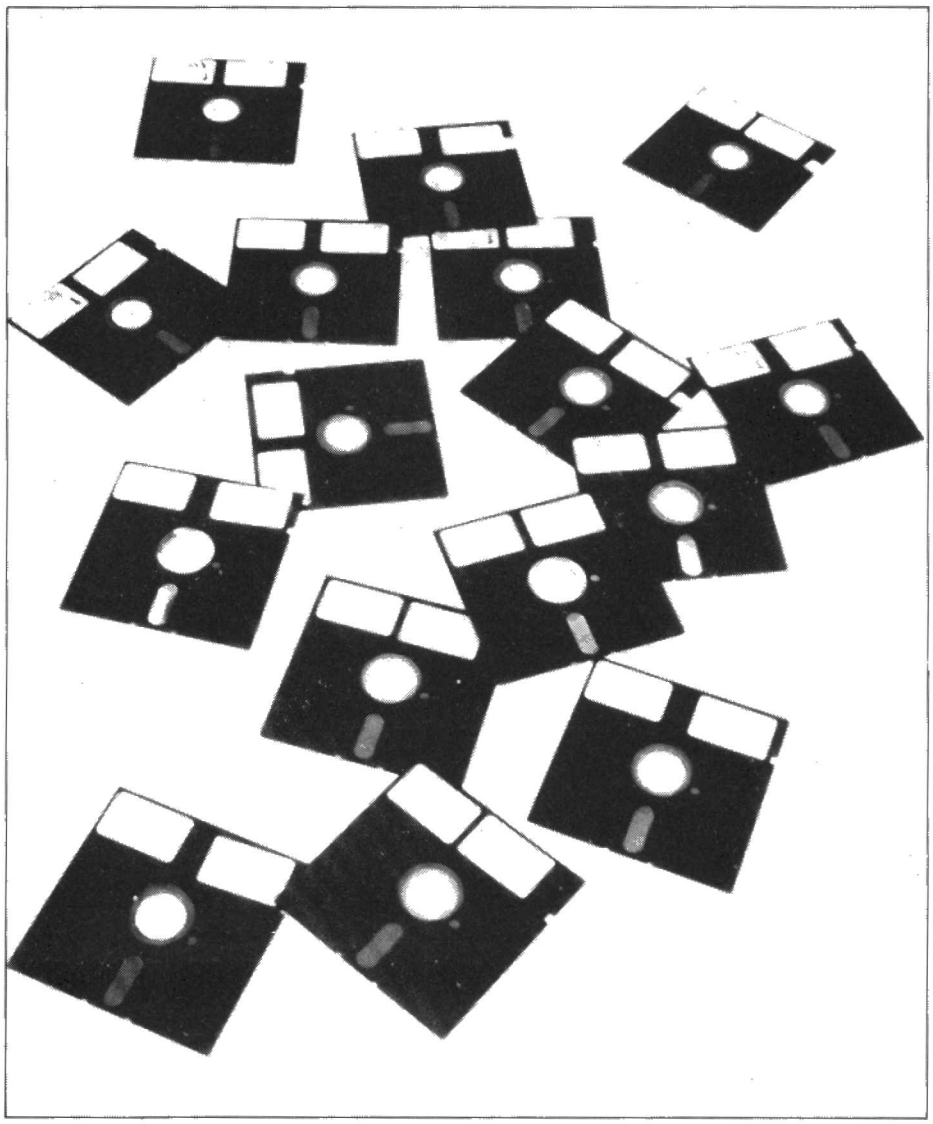
PROCselect_directory allows the user to choose which directories to add to the catalogue of programs. You can read all of the disks which is the default setting. Otherwise simply type in the directory letters eg +– for DiscDoctor users. If it is necessary to read all of the directories again then enter "ALL" at this point.

You are now invited to insert the disk that is ready for reading. Following the invitation to press any key, you can press * if you wish to perform a *command. This allows you to *CAT. the disk, to *SWAP or *ALT the disk for those using double catalogues on disk. Finally you can at this stage retitle the disk if it is necessary.

PROCget_cat copies the first two sectors of the disk into a buffer at &1200. For some reason this operation doesn't always work first time. A disk error message is given and it tries again. It is usually successful the second time.

PROCread_cat reads the information which is on sectors 0 & 1 which has now been loaded into the buffer mentioned above.

It first checks that the disk has not already been catalogued by running through the stored list of disk titles. If it finds the same title it prints a warning message and exits from the routine. This also



LISTING 1

```
  100 REM ********************
  110 REM * Disc Cataloguer *
  120 REM *       by        *
  130 REM *  M.E.Williams   *
  140 REM ********************
  150
  160 MODE7
  170 PROCsetup
  180
  190 REM *** Main loop ***
  200
  210 REPEAT
  220   PROCcatalogue
  230   PROCsave
  240   PROCdisplay
  250   PROCprint
  260   IF NOT full_up THEN PROCheader("Add to the catalogue?")
  270   UNTIL full_up OR NOT FNy_n
  280 PROCsave
  290 PROClarge("Bye...."):END
  300 *DIR $:REM return to default directory
  310 END
  320
  330 DEFPROCsetup
  340 VDU23;8202;0;0;0;:REM cursor off
  350 osword=&FFF1:oscli=&FFF7
  360 DIMrdblock% 11:buf1%=&1200:buf2%=&1300
  370 print_flag=FALSE:dir$="ALL":full_up=FALSE:t$=STRING$(12," ")
  380 sct=0:tnum=0:siznumwrv=&22
  390 drive%=0:disc%=0:prognum%=1
  400 DIM file$(800),P%(800),title_pointer%(800),title$(50)
  410 PROCload_data
  420 ENDPROC
  430
  440 DEFPROCcatalogue
  450 REPEAT
  460   PROCselect_directory
  470   PROCheader("Insert disc and press any key"):J%=GET
  480   IF J%<>42 THEN 500 ELSE PROCoscli
  490   PROCspace:GOTO 520
  500   PROCget_cat
  510   PROCread_cat
  520   PROCheader("Another disc? ")
  530   UNTIL NOT FNy_n OR full_up
  540 ENDPROC
  550
  560 DEFPROCorder:LOCAL K%,N%,Z%
  570 Z%=prognum%:N%=0
  580 REPEAT:N%=N%+1
  590   UNTIL prog$<file$(P%(N%)) OR N%=prognum%
  600 IF N%=prognum% THEN P%(prognum%)=Z%:ENDPROC
  610 FOR K%=prognum%-1 TO N% STEP-1
  620   P%(K%+1)=P%(K%)
  630   NEXT
  640 P%(N%)=Z%:ENDPROC
  650
  660 DEFPROCoscli
  670 PROCheader("*Command?")
  680 PRINT''TAB(11);
  690 PROCinput(20,31,128)
  700 CLS
  710 $&1200=In$
  720 X%=0:Y%=&12:CALLoscli
  730 ENDPROC
  740
  750 DEFPROCget_cat
  760 ?rdblock%=drive%:rdblock%(1)=buf1%:rdblock%?5=7:rdblock%?6=&50
  770 tr=rdblock%+7:sr=tr+1:cr=tr+2:ern=tr+3
  780 ?tr=tnum:?sr=sct:?cr=siznumwrv:?ern=0
  790 X%=rdblock%:Y%=rdblock% DIV 256:A%=&7F:CALLosword
  800 IF ?ern<>0 PRINT'':PROCLarge("Disc error..trying again"):GOTO 760
  810 ENDPROC
  820
  830 DEFFNtitle:LOCAL J%
  840 t$=""
  850 FOR J%=0 TO 7:t$=t$+CHR$(buf1%?J%):NEXT
  860 FOR J%=0 TO 3:t$=t$+CHR$(buf2%?J%):NEXT
  870 =t$
  880
  890 DEFPROCread_cat
  900 pointer%=buf1%+7:VDU15
  910 already=FALSE:FOR J%=1 TO disc%:IF title$(J%)=FNtitle THEN already=TRUE:NEXT ELSE
NEXT
  920 IF already THEN VDU7:PROCheader("Already catalogued!"):PROCwait(200):ENDPROC
  930 n_of_progs%=buf2%?5/8:IF n_of_progs%=0 THEN 1110:REM empty disc
  940 disc%=disc%+1:IF disc%=50 THEN full_up=TRUE
  950 title$(disc%)=FNtitle:K%=0
  960 REM loop to read catalogue and order alphabetically
  970 FOR J%=1 TO n_of_progs%
  980   IF dir$="ALL" THEN 1000:REM otherwise check if chosen directories
  990   d$=CHR$((pointer%?8)AND &7F):IF INSTR(dir$,d$)=0 THEN pointer%=pointer%+8:GOTO
1090
 1000   prog$="":K%=0
 1010   REPEAT:K%=K%+1:prog$=prog$+CHR$(pointer%?K%)
 1020     UNTIL (CHR$(pointer%?K%)=" " OR K%=7)
 1030   pointer%=pointer%+8
 1040   IF LEFT$(prog$,1)="!" THEN 1090:REM avoid !BOOTs
 1050   file$(prognum%)=prog$:title_pointer%(prognum%)=disc%
 1060   PRINTTAB(16)prog$
 1070   PROCorder
 1080   prognum%=prognum%+1:IF prognum%=501 THEN full_up=TRUE:J%=n_of_progs%
 1090   NEXT
 1100 IF K%=0 THEN disc%=disc%-1:REM no progs in chosen directories
 1110 PROCfree:PROCspace
 1120 PROCspace
 1130 ENDPROC
 1140
 1150 DEFPROCselect_directory
 1160 PROCheader("Directories = "+dir$)
 1170 PRINT''TAB(10);:PROCLarge("Directories OK?")
 1180 IF FNy_n THEN ENDPROC
 1190 PROCheader("Enter the directory letters.")
 1200 PRINTTAB(14,6);:PROCinput(10,32,126)
 1210 dir$=In$
 1220 ENDPROC
 1230
 1240 DEFPROCdisplay
 1250 REPEAT
 1260   PROCheader("Sorted alphabetically?")
 1270   IF FNy_n THEN sort=TRUE ELSE sort=FALSE
 1280   PROCheader("Disc Catalogue")
 1290   VDU14:PROCwindow(0,3,39,22)
 1300   IF print_flag THEN VDU2,15:REM pageing off
 1310   FOR J%=1 TO prognum%
 1320     IF sort THEN JJ%=P%(J%) ELSE JJ%=J%
 1330     PRINTTAB(8)file$(JJ%);
 1340     temp$=title$(title_pointer%(JJ%)):IF NOT print_flag THEN 1390
 1350     stripped$="":REM remove teletext colours
 1360     FOR K%=1 TO LEN(temp$)
 1370       IF ASC(MID$(temp$,K%,1))<128 THEN stripped$=stripped$+MID$
(temp$,K%,1)
 1380       NEXT:temp$=stripped$
 1390     PRINTTAB(22)temp$
 1400     NEXT
 1410   VDU3:print_flag=FALSE
 1420   PROCspace
 1430   VDU26:REM reset window
 1440   PROCheader("View again?")
 1450   UNTIL NOT FNy_n
 1460 ENDPROC
 1470
 1480 DEFPROCsave
 1490 PROCheader("Save data onto disc?")
 1500 IF NOT FNy_n THEN ENDPROC ELSE PROCget_name
 1510 X%=OPENOUT(file$)
 1520 PRINT#X%, prognum%
 1530 PRINT#X%,disc%
 1540 FOR J%=1 TO prognum%
 1550   PRINT#X%,file$(J%)
 1560   PRINT#X%,P%(J%)
 1570   PRINT#X%,title_pointer%(J%)
 1580   NEXT
 1590 FOR J%=1 TO disc%
 1600   PRINT#X%,title$(J%)
 1610   NEXT
 1620 CLOSE #X%
 1630 ENDPROC
 1640
 1650 DEFPROCload_data
 1660 PROCheader("Load data from disc?")
 1670 IF NOT FNy_n THEN ENDPROC ELSE PROCget_name
 1680 X%=OPENIN(file$)
 1690 IF X%=0 THEN PROCheader("Not Found!"):VDU7:PROCwait(300):GOTO 1660
 1700 PROCheader("Loading...")
 1710 INPUT#X%, prognum%
 1720 INPUT#X%,disc%
 1730 FOR J%=1 TO prognum%
 1740   INPUT#X%,file$(J%)
 1750   INPUT#X%,P%(J%)
 1760   INPUT#X%,title_pointer%(J%)
 1770   NEXT
 1780 FOR J%=1 TO disc%
 1790   INPUT#X%,title$(J%)
 1800   NEXT
 1810 CLOSE #X%
 1820 PROCdisplay
 1830 PROCheader("Add to the file?")
 1840 IF FNy_n THEN ENDPROC
 1850 PROCprint
 1860 CLS:END
 1870
 1880 DEFPROCget_name
 1890 PROCheader("What name?")
 1900 PRINTTAB(15,6);:PROCinput(9,32,123)
 1910 file$=In$
 1920 ENDPROC
 1930
 1940 DEFPROCprint
 1950 PROCheader("Printed Catalogue?")
 1960 IF NOT FNy_n THEN ENDPROC ELSE print_flag=TRUE
 1970 VDU2
 1980 PRINTTAB(13)"DISC CATALOGUE"
 1990 PRINTTAB(13)"=============="
 2000 VDU3
 2010 PROCdisplay
 2020 GOTO1950
 2030 ENDPROC
 2040
 2050 DEFPROCspace
 2060 PRINT'"          PRESS SPACE  ";
 2070 REPEAT:UNTIL INKEY-99
 2080 ENDPROC
 2090
 2100 DEFPROCfree
 2110 nfs%=(buf2%?&E AND 7)*256+buf2%?&F+(buf2%?&E AND &30)
      *16+buf2%?&D-(buf2%<>0)
 2120 IF buf2%?5/8=0 THEN nfs%=2:REM next free sector
 2130 free_sectors=(buf2%?5 AND 3)*256+buf2%?4-nfs%
 2140 PRINT'TAB(5)free_sectors;" free sectors."
 2150 ENDPROC
 2160
 2170 DEFPROCwindow(SX,SY,W,H):REM start,width,height
 2180 VDU28,SX,SY+H-1,SX+W-1,SY
 2190 ENDPROC
 2200 REM*****  MODE 7 procedures       *****
 2210 DEFPROCcenter(A$,vpos)PRINTTAB((20-LENA$DIV2),vpos);A$:ENDPROC
 2220 DEFPROCheader(A$)VDU12:FORI%=0 TO
1:PRINTCHR$&81;CHR$&9D;CHR$&83;CHR$&BD;:PROCcenter(A$,VPOS):NEXT:ENDPROC
 2230 DEFPROClarge(A$)FORI=0TO1:PROCcenter(CHR$&8D+A$,VPOS):NEXT:ENDPROC
 2240
 2250 DEFFNy_n:LOCALI%:REPEAT:I%=INSTR("Y.y,N,n",GET$):UNTILI%
 2260 IFI%<3 =TRUE ELSE=FALSE
 2270
 2280 REM*****  General purpose input  *****
 2290 DEFPROCinput(L%,Lo%,Hi%):REM string length<=L% ASC betweenLo% and Hi%
 2300 LOCALK%,Z%:K%=0:In$=""
 2310 PRINTSTRING$(L%,"_");STRING$(L%+1,CHR$(8));" ";:*FX15,1
 2320 REPEAT:Z%=GET:IFZ%=127ANDK%>0THENPROCdel
 2330   IFZ%>Lo%ANDK%<L%ANDZ%<Hi%THENPROCadd
 2340   UNTIL(Z%=13ANDK%>0)OR(L%=1ANDK%=1):ENDPROC
 2350 DEFPROCdel:K%=K%-1:In$=LEFT$(In$,K%):PRINTCHR$Z%;" ";CHR$(8);:ENDPROC
 2360 DEFPROCadd:K%=K%+1:In$=In$+CHR$(Z%):VDUZ%:ENDPROC
 2370
 2380 DEFPROCwait(t%):TIME=0:REPEAT UNTIL TIME>t%:ENDPROC
```

happens if an attempt is made to catalogue two unnamed disks. As mentioned above, at the insert disk stage it is possible to carry out a *TITLE. It next checks for an empty disk and gives an appropriate message.

Finally it reads through the list of programs looking for those in the chosen directories. Line 1040 makes sure that you don't end up with dozens of !BOOTs on the list. As the programs are read in they also appear on the screen.

As a bonus, after the names have been printed the number of free sectors is found by PROCfree and given on screen. This is only the number of sectors after the last file on the disk and does not include sectors which might be released by compacting the disc.

As the number of programs builds up you will notice that they slow down somewhat. This is due to PROCorder. As each program is read in it is stored into the file$ array. A second array stores the number of the disk titles within the title$ array. This takes less space than storing the title name many times. The slowing down is due to the shuffling that is done on the pointer array P%. This array stores the number of the programs in file$, but in alphabetical order. So when a new program has to be entered the procedure finds where in the pointer array the new program has to go. Then it shifts all the others down one position (lines 610-630). Above a couple of hundred programs this simple method is rather slow.

Having catalogued the disk you are invited to go on to the next disk by cycling through the above procedures. The catalogue can be viewed once each disk is entered; it will group by disk or in alphabetical order. Coloured disk titles are very useful.

"Using the program will make life much easier".

PROCsave

The catalogue can then be saved to disk under a suitable name. Thus a series of catalogues such as 'games', 'utilities', 'tunes' and so on, can be saved.

PROCdisplay

As well as displaying the list on screen, this procedure handles the printer. This is where coloured titles give some problems. The mode 7 colour codes do horrible things to printers. So lines 1350-1380 have the job of stripping them out before sending the disk titles to the printer. Some customising for your own printer could well go in here. The rest of the procedures set up windows, print headers, check the inputs and so on.

So there you have it. Using the program should make life easier and much less frustrating. Visiting friends can be handed 'menus' to choose from. Forgotten gems are likely to resurface and order will be imposed upon chaos. Entropy can be defeated and this program shows you how!

Postscript

The idle, as well as the gentleman who once wrote me that he found it difficult to hold the magazine in one hand, magnifying glass in another and type all at the same time, can be provided with the program on 40-track disk for £5.50 including p&p.

Cheques to: *M. E. Williams, 11 Cressy Road, London NW3 2NB.*

Golden rules for efficient program storage

(i) All discs should be titled differently. If possible, coloured titles should be used.

(ii) Full use should be made of the directory facility. Many games for example come in multiple parts. By putting the first part into a G directory, or + directory, and then having the subsidiary parts in a second directoy (with corresponding change of directory by the first part), you only have to catalogue the main directory and thereby avoid a lot of unwanted titles in your master catalogue.

RANDOM ACCESS

Adam Denning got the random access filing system project underway last month with three BASIC procedures designed to assemble code for supervisory operations. This month he enhances the facilities offered by moving into machine code.

The major project to design a random access filing system based on a record map which holds the details of all records on the disc got underway last month. To get the system started, three BASIC procedures were written to assemble code for supervisory operations. However in order to enhance the power of this system we will move into machine code, so all the programs from now on are shown in a form suitable for the System ADE assembler. This, of course, can easily be converted to BBC BASIC assembler.

The three routines which appeared last month, PROCinit, PROCstart and PROCfinish, have all been converted to dispense with BASIC, and the PROCread_record routine has been added. Note that, as these are now entirely in machine code they cannot be called from BASIC as if they were procedures – it is necessary to make a note of each address so that they can be called at the relevant places.

The first three routines, starting at INIT, START and FINISH are all self-contained and need no parameters, except that the equates for MAPLOC, RECFLE, RECLEN and RECMAP may need altering and a different ORG address may be chosen.

These three routines are essentially the same as the BASIC procedures shown last month, but the free zero page locations for &70 upwards are used to hold various data. RDRECD is a new routine and needs explanation. Remember that START loaded the record map into memory starting at MAPLOC, and put the file handle returned by OSFIND for RECFLE into the zero page location HANDLE. Before calling this routine make sure that the TWO BYTE location RECNUM holds the number of the record which you want to load, and the four bytes starting at OSBLOK + 1 holds the address to which you want the record to be loaded. This can be done by using

```
!(OSBLOK+1)=record_load_address%
```

or a similar construct.

RDRECD works by first taking the number of records in the record file (remember that this is held in the first two bytes of the record map file) and storing this value in MAXREC. Then check that this is not zero – after all, if there are zero records in our file then our record will never be located. If this is the case then the routine jumps to BADREC. Next the address of the start of the record map is stored in CURECD and each entry is scanned until either our record is found or the memory-resident file is exhausted. Each time around the loop, CURECD is incremented by two, thus making it point to the next entry in the record map, and MAXREC is decremented. If MAXREC becomes zero then the file is exhausted so we leave at BADREC. In both the instances when the program jumps to BADREC, the condition was 'zero flag set', so we return straightaway to the calling routine with the zero flag set, indicating failure. We'll look at ways of interpreting this next month.

If our record is found then the program jumps to GOTREC. Note that a two byte value can be checked for being zero by loading one byte into A and then logical ORing it with the second byte. Only 0 OR 0 will return a result of zero, so only this condition will result in the zero flag being set. Once at GOTREC a new sequential pointer for the record file must be generated by multiplying the record number by the record length. The record length is a constant defined earlier in the program as RECLEN, and the record number is currently in RECNUM. There are various multiplication algorithms but we'll plump for a simple repetitive addition method. Notice also that the code here multiplies two 16-bit numbers and assumes a 16-bit result, so it must be ensured that either our maximum pointer value is less than 65536 or that the code is altered to produce a 32-bit result. This is extremely tedious!

The multiplication routine first zeroes CURECD so that it can use it as a cumulative total, and then it checks RECNUM to see if it is zero. If this is the case then the routine finishes. This therefore covers the 'zero case', where we are looking for record number zero, as the routine will terminate immediately.

Now that we have the new sequential pointer in CURECD (at GOTPTR) the operating system routine OSGBPB (Operating System Get Bytes Put Bytes) must be used to read the record into memory. This requires XY to point to a control block, for which we use OSBLOK, and this control block must be set up as follows:

OSBLOK + 00:	file handle
OSBLOK + 01:	load address for record (this must be present before RDRECD is called)
OSBLOK + 08:	number of bytes to be read
OSBLOK + 09:	new sequential pointer

The accumulator must be loaded with 3 for 'read bytes from file using the new sequential pointer'. It must be assumed that the call has gone well as our record structure is not going to get corrupted by our routines. At the end of the routine, we load A with 255 just to reset the zero flag so that upon return to the calling routine we know that everything has gone well.

The next installment of Random Access will describe the WRTREC routine (equivalent to PROCwrite_record) and the surrounding code which calls each routine and interprets the result. Finally we'll be able to incorporate the whole block of code into a program which constructs records and reads them back. ▶

"In order to enhance the power of our random access filing system we must leave BASIC and move into machine code".

LISTING 1

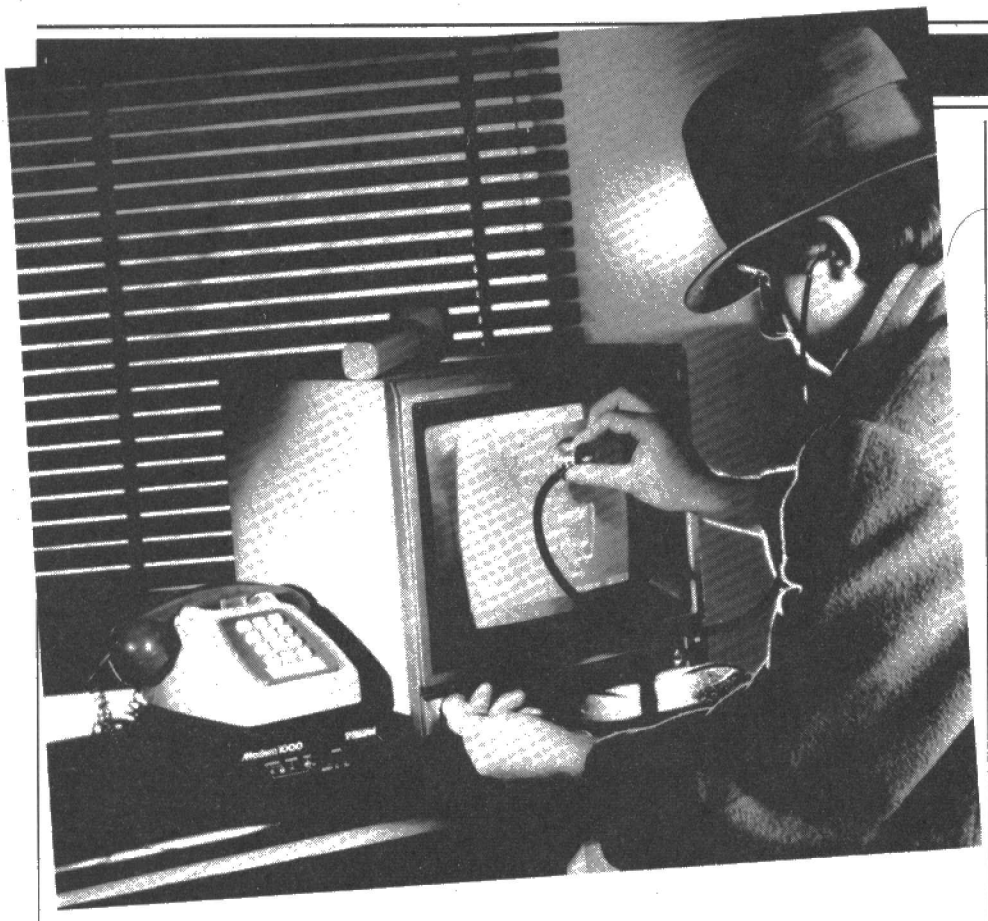
```
0000:          1 * Random Access filing system using a record map
0000:          2 * By Adam Denning 10th December 1984
0000:          3 * (C) 1984 Adam Denning
0000:          4
0070=          5 RECNUM  EQU     $70     ; !! TWO BYTE location !!
                                           record being searched for
0072=          6 CURECD  EQU     $72     ; current record number in search
0074=          7 MAXREC  EQU     $74     ; number of records in map
0076=          8 HANDLE  EQU     $76     ; file handle for record_file
0000:          9
0100=         10 RECLEN  EQU     $100    ; set this to your record size
7000=         11 MAPLOC  EQU     $7000   ; set this to wherever you want
                                           the map to go
0000:         12
FFCE=         13 OSFIND  EQU     $FFCE
FFD1=         14 OSGBPB  EQU     $FFD1
FFD4=         15 OSBPUT  EQU     $FFD4
FFDA=         16 OSARGS  EQU     $FFDA
FFDD=         17 OSFILE  EQU     $FFDD
0000:         18
0000:         19         ORG     $6000          ;or wherever you choose
6000:         20
6000:         21 * The initialisation routine (PROCinit)
6000:         22
6000:A980     23 INIT    LDA #   $80
6002:A2DA     24         LDX     #>RECFLE
6004:A060     25         LDY     #<RECFLE
6006:20CEFF   26         JSR     OSFIND
6009:AB       27         TAY
600A:F01C     28         BEQ     ERROR
600C:A900     29         LDA     #0
600E:20CEFF   30         JSR     OSFIND
6011:A980     31         LDA     #$80
6013:A2E2     32         LDX     #>RECMAP
6015:A060     33         LDY     #<RECMAP
6017:20CEFF   34         JSR     OSFIND
601A:AB       35         TAY
601B:F00B     36         BEQ     ERROR
601D:A900     37         LDA     #0
601F:20D4FF   38         JSR     OSBPUT
6022:20D4FF   39         JSR     OSBPUT
6025:4CCEFF   40         JMP     OSFIND
6028:         41
6028:00       42 ERROR   BRK
6029:00       43         BRK
602A:46696C   44         ASC     "File error!"
6035:00       45         BRK
6036:         46
6036:         47 * The start-up routine (PROCstart)
6036:         48
6036:A9C0     49 START   LDA     #$C0
6038:A2DA     50         LDX     #>RECFLE
603A:A060     51         LDY     #<RECFLE
603C:20CEFF   52         JSR     OSFIND
603F:AB       53         TAY
6040:F0E6     54         BEQ     ERROR
6042:48       55         PHA
6043:A2E2     56         LDX     #>RECMAP
6045:8EC860   57         STX     OSBLOK
6048:A260     58         LDX     #<RECMAP
604A:8EC960   59         STX     OSBLOK+1
604D:A900     60         LDA     #0
604F:8DCE60   61         STA     OSBLOK+6
6052:8DCF60   62         STA     OSBLOK+7
6055:8DD060   63         STA     OSBLOK+8
6058:8DD160   64         STA     OSBLOK+9
605B:A900     65         LDA     #>MAPLOC
605D:8DCA60   66         STA     OSBLOK+2
6060:A970     67         LDA     #<MAPLOC
6062:8DCB60   68         STA     OSBLOK+3
6065:A900     69         LDA     #0
6067:8DCC60   70         STA     OSBLOK+4
606A:8DCD60   71         STA     OSBLOK+5
606D:8DCE60   72         STA     OSBLOK+6
6070:8DCF60   73         STA     OSBLOK+7
6073:8DD060   74         STA     OSBLOK+8
6076:8DD160   75         STA     OSBLOK+9
6079:A2C8     76         LDX     #>OSBLOK
607B:A060     77         LDY     #<OSBLOK
607D:A9FF     78         LDA     #$FF
607F:20DDFF   79         JSR     OSFILE
6082:68       80         PLA
6083:8576     81         STA     HANDLE
6085:60       82         RTS
6086:         83
6086:         84 * The finalisation routine (PROCfinish)
6086:         85
6086:A476     86 FINISH  LDY     HANDLE
6088:A900     87         LDA     #0
608A:20CEFF   88         JSR     OSFIND
608D:A2E2     89         LDX     #>RECMAP
608F:8EC860   90         STX     OSBLOK
6092:A260     91         LDX     #<RECMAP
6094:8EC960   92         STX     OSBLOK+1
6097:A900     93         LDA     #0
6099:8DCA60   94         STA     OSBLOK+2
609C:8DCB60   95         STA     OSBLOK+3
609F:8DCC60   96         STA     OSBLOK+4
60A2:8DCD60   97         STA     OSBLOK+5
60A5:8DCE60   98         STA     OSBLOK+6
60A8:8DCF60   99         STA     OSBLOK+7
60AB:8DD060  100         STA     OSBLOK+8
60AE:8DD160  101         STA     OSBLOK+9
60B1:8DD460  102         STA     OSBLOK+12
60B4:8DD560  103         STA     OSBLOK+13
60B7:A200    104         LDX     #>MAPLOC
60B9:A070    105         LDY     #<MAPLOC
60BB:8ED260  106         STX     OSBLOK+10
60BE:8CD360  107         STY     OSBLOK+11
60C1:A2C8    108         LDX     #>OSBLOK
60C3:A060    109         LDY     #<OSBLOK
60C5:4CDDFF  110         JMP     OSFILE
60C8:        111
60C8:000000  112 OSBLOK  DS      $12
60DA:        113
60DA:        114 * The file names
60DA:        115
60DA:726563  116 RECFLE  STR     "records" ; or whatever you choose for 'record_file'
60E2:        117
60E2:726563  118 RECMAP  STR     "rec_map" ; or whatever you choose for 'record_map'
60EA:        119
60EA:        120 * The 'read a record' routine (PROCread_record)
60EA:        121 * Entered with desired record number in RECNUM and the
                  address to which the
60EA:        122 * record is to be loaded in !(OSBLOK+1). Returns with
                  the zero flag set if
60EA:        123 * the record could not be found, reset otherwise.
60EA:        124
60EA:AE0070  125 RDRECD  LDX     MAPLOC
60ED:AC0170  126         LDY     MAPLOC+1
60F0:8674    127         STX     MAXREC
60F2:8475    128         STY     MAXREC+1
60F4:8A      129         TXA
60F5:0575    130         ORA     MAXREC+1
60F7:F034    131         BEQ     BADREC
60F9:A202    132         LDX     #>MAPLOC+2
60FB:A070    133         LDY     #<MAPLOC+2
60FD:8672    134         STX     CURECD
60FF:8473    135         STY     CURECD+1
6101:A000    136 GETREC  LDY     #0
6103:B172    137         LDA     (CURECD),Y
6105:C570    138         CMP     RECNUM
6107:D007    139         BNE     NOTREC
6109:C8      140         INY
610A:B172    141         LDA     (CURECD),Y
610C:C571    142         CMP     RECNUM+1
610E:F01E    143         BEQ     GOTREC
6110:18      144 NOTREC  CLC
6111:A572    145         LDA     CURECD
6113:6902    146         ADC     #2
6115:8572    147         STA     CURECD
6117:A573    148         LDA     CURECD+1
6119:6900    149         ADC     #0
611B:8573    150         STA     CURECD+1
611D:A574    151         LDA     MAXREC
611F:08      152         PHP
6120:C674    153         DEC     MAXREC
6122:28      154         PLP
6123:D002    155         BNE     NODECH
6125:C675    156         DEC     MAXREC+1
6127:A574    157 NODECH  LDA     MAXREC
```

LISTING 1 – Continued

```
6129:0575   158         ORA     MAXREC+1
612B:D0D4   159         BNE     GETREC
612D:60     160 BADREC  RTS         ; returns with zero flag set if no such record
612E:       161
612E:       162 * Reaches this point when (CURECD) points to RECNUM
612E:       163 * Must now multiply RECNUM by RECLEN and set PTR to this value
612E:       164 * Being thick, I've gone for the simplest alogrithm:
612E:       165
612E:A900   166 GOTREC  LDA     #0
6130:8572   167         STA     CURECD
6132:8573   168         STA     CURECD+1
6134:A570   169 GETPTR  LDA     RECNUM
6136:0571   170         ORA     RECNUM+1
6138:F01A   171         BEQ     GOTPTR
613A:18     172         CLC
6138:A900   173         LDA     #>RECLEN
613D:6572   174         ADC     CURECD
613F:8572   175         STA     CURECD
6141:A901   176         LDA     #<RECLEN
6143:6573   177         ADC     CURECD+1
6145:8573   178         STA     CURECD+1
6147:A570   179         LDA     RECNUM
6149:08     180         PHP
614A:C670   181         DEC     RECNUM
614C:28     182         PLP
614D:D0E5   183         BNE     GETPTR
614F:C671   184         DEC     RECNUM+1
6151:18     185         CLC
6152:90E0   186         BCC     GETPTR
6154:A672   187 GOTPTR  LDX     CURECD
6156:A473   188         LDY     CURECD+1
615B:8ED160 189         STX     OSBLOK+9
615B:8CD260 190         STY     OSBLOK+10
615E:A476   191         LDY     HANDLE
6160:8CC860 192         STY     OSBLOK
6163:A200   193         LDX     #>RECLEN
6165:A001   194         LDY     #<RECLEN
6167:8ECD60 195         STX     OSBLOK+5
616A:8CCE60 196         STY     OSBLOK+6
616D:A900   197         LDA     #0
616F:8DCF60 198         STA     OSBLOK+7
6172:8DD060 199         STA     OSBLOK+8
6175:8DD360 200         STA     OSBLOK+11
6178:8DD460 201         STA     OSBLOK+12
617B:       202
617B:       203 * REMEMBER - !(OSBLOK+1) MUST BE INITIALISED WITH THE LOAD ADDRESS
617B:       204
617B:A903   205         LDA     #3
617D:A2C8   206         LDX     #>OSBLOK
617F:A060   207         LDY     #<OSBLOK
6181:20D1FF 208         JSR     OSGBPB
6184:A9FF   209         LDA     #$FF
6186:60     210         RTS         ; return with zero flag reset - success!!
6187:       211
BADREC  612D        CURECD  0072        ERROR   6028        FINISH  6086
GETPTR  6134        GETREC  6101        GOTPTR  6154        GOTREC  612E
HANDLE  0076        INIT    6000        MAPLOC  7000        MAXREC  0074
NODECH  6127        NOTREC  6110        OSARGS  FFDA        OSBLOK  60C8
OSBPUT  FFD4        OSFILE  FFDD        OSFIND  FFCE        OSGBPB  FFD1
RDRECD  60EA        RECFLE  60DA        RECLEN  0100        RECMAP  60E2
RECNUM  0070        START   6036

 9775 Bytes free - No error(s)

BASIC
```



HACKERS EXPOSED!

E&CM's own super hacker, who shall remain nameless for obvious reasons, looks at the art of hacking and comes up with some useful tips for database managers looking to improve security.

A well known public company recently installed a £3m computer system with terminals throughout their London office. But employees wishing to work from home, or to be able to keep track of situations while travelling, will be disappointed in the new system. Company policy states that the new computer will not be linked, by any means, to remote terminals either through the public network or even on private land lines. This is what's known as a fear of hackers.

In the past few years, Hacking has become the word to describe people who, armed only with a micro and a modem, break into other people's computers through the telephone network.

Prestel, British Telecom's viewdata system, is probably the most widely known network system in the UK. Ironically, it probably gained much greater public recognition when, a couple of months ago, a hacker managed to crack the system and read the contents of Prince Philip's mailbox. Although, I'm reliably informed, this contained nothing of interest, it certainly made the newspapers. And not just in this country, either. The story was given pride of place on the front page of La Nazione – 500 words of bold type for all of Italy to read.

Most hacks take place through negligence on the part of the system operator. Prestel, for example, always told us to change our passwords regularly and not to use ones which could be easily guessed. Yet when the big Prestel hack took place, the unnamed hacker had access to the system manager area and could interrogate the password files. What was interesting was that many Prestel employees had passwords of 0000, or had used their first names. And these passwords were a year or two old! Even 10-digit ID numbers are not as secure as they should have been. An account existed on Prestel with an ID of 7977777777 and a password of 7777.

After Prestel was exposed, all information providers had their 10-digit IDs changed. Yet their 4-character editing passwords remained the same. This extra level of security, an additional password which is required when actually changing information as opposed to just reading it, was overlooked in the spring clean. What Prestel forgot was that the Bulk Update system (used for high-speed 1200 baud updating of pages from a pre-prepared disk file) doesn't ask for a 10-digit ID; just the editing password! No, weeks after the hack took place, the system was still fully open. And, if you're wondering how the system manager ID was discovered by a hacker, it was left on a BT-only area of the database, accessible by using an ID of 2222222222 and a password of 1234.

Other systems are not even as secure as this. A travel agent in my local high street recently left their viewdata terminal on overnight, with the menu of phone numbers on full display. A midnight stroll with a pair of opera glasses revealed phone numbers of half a dozen private viewdata systems, used by various tour operators.

I dialled Thomas Cook. The system requested a 6-digit ID and a 4-character password. I tried 777777 and 7777. It worked. I logged off, and the system didn't even clear down the line. I was returned to the log on page to try another ID. This time, I tried 999999 and 9999. Incorrect password. 1234? Second time lucky. It really is as easy as that.

As one story said, after the Prestel incident, it wasn't that the robbers broke into the bank; more that the manager left the keys under the mat.

Incidentally, although Information Providers have now had their IDs changed, the 45,000 users like you and I have not. If you are one of these, ask Prestel to change your ID number tomorrow.

System security

So, what can be done to make a system secure? One company not to go for advice is ICL. This computer firm took out around 150 accounts on Telecom Gold, an electronic mail service. Almost all their IDs (coded ICL001 to ICL150) had a password of ICL. Needless to say, these were soon hacked and removed.

"It is surprising how much some computers help a hacker once the system's 'phone number is known".

Any computer system must keep careful logs of usage. If an ID suddenly starts to become used more often than usual, or at strange times of the day or night, the operator should be informed and the owner of that ID asked to verify the situation. If a user gets a password wrong on a large number of occasions, the same

action should be taken. Most importantly, this log should not be accessible over the phone system. It must be available only from the system operator's console from inside the building itself.

Prestel's security system is such that any strange usage patterns on an ID result in a mailbox message being automatically generated by the system and sent to the security manager's personal ID. All that a hacker needs to do to remove all traces of his nocturnal tour through the system files, is to log on as the security manager and erase the contents of that mailbox.

The line should be cleared after every call, whether or not it is successful. The computer must "hang up" the phone, and not simply represent the caller with the logon page. Otherwise, hackers can write simple programs to spend a whole night trying random IDs and passwords, all for the cost of one call. If a password is entered incorrectly, just once, a further call should be necessary.

Users should be able to change both their IDs and passwords as required. Passwords should be at least 6 characters in length (as now required by Gold) and should be alphanumeric.

It is surprising how much a computer can help a hacker once he has found the phone number of the system. On some networks, typing HELP while the computer is asking for an ID will produce the the procedure for logging on, and the format and syntax for IDs and passwords.

It is all very well to be user friendly, but make sure that you're not hacker-friendly too.

When a system asks for an ID and password, it should request both before saying whether or not a correct entry has been made. Some systems, for example, ask for the ID and will only ask for a password if the ID is valid. If it isn't, the computer prints INVALID ID and asks for it again. This means that a hacker knows where he is going wrong.

Information directories are a good idea, but there are some entries which should not be made public knowledge. Log on to Hostess, for example, the BT host computer for PSS. Call up the PSS directory and, at the top of the list, you'll find the PSS number for the Atomic Energy Research Establishment at Harwell. Quite why this is in a public directory, I don't know.

"It's not that robbers break into the (data) bank more that the manager often leaves the key under the mat . . ."

Reasons for hacking

What can be gained from hacking? Information, mainly. Most hacks in this country are fairly amateur in nature. Sure, there are the pros who will break into anything if you pay them enough. A well known tabloid recently offered me lots of money to obtain details of a well known TV personality's Barclaycard account.

Usually, though, hackers do it for fun. Because it's there. Damage is usually slight as far as national security goes, though some hacks can be costly. Computing time is not cheap, and accessing something like a CRAY-1 while charging the session to someone else is fairly mean.

Serious hacking is rare, though minor hacks are commonplace on Prestel. With some expensive telesoftware already on the system and more on the way, don't follow Prestel's example – change your passwords regularly.

CompuNet hackers are, in a way, luckier than Prestel ones. You are allowed to upload programs or information on to CompuNet and then to put a price on that page. If someone looks at it, you get half the price credited to your account while CompuNet gets the other. If you put up a blank page with a price tag of £5 and then repeatedly access that page using a hacked ID, you are, as one well-known criminal might put it, in for a nice little earner.

READY . . . STEADY . . . GO!

Paul Beverley expands his race track timing system for the BBC micro with a facility for monitoring the performance of two cars. Further enhancements include a warning lamp cluster driven from the computer's printer port.

The technique used for a single track was to sit and wait for the car to go over the detector and then respond by calculating the speeds and displaying results for that lap. The timing of two cars is more complex because the computer cannot do two things at once even though it may seem at times that it can. Therefore it is possible that, while the computer is responding to the fact that one of the cars has just gone over its detector, the other car might slip over its detector unnoticed.

However, there is actually plenty of time to do the calculations while the cars are going round the track. In fact the computer spends the majority of its time just waiting for the cars to come. The only difficulty is that, while the computer is doing some calculations, it is unable to watch what is happening on the track for more than a few centiseconds and so one of the cars may get through undetected.

In order to both calculate and display the results during the course of the race it becomes necessary to go into machine code and to start using interrupts. This will be dealt with next month.

It would seem then, that it is preferable not to do any displaying or calculating during the course of the race, or at least only a very little, very quickly. It is possible to do the single track timings in BASIC without resorting to machine code, and, with care, also to do the two track version in BASIC. However, it is asking too much to measure the instantaneous speeds of the cars as they go over the detectors as with the single track program.

The Hardware

For the benefit of those who missed last month's article, the hardware is shown again in **Figure 1.** It consists of two TIL100 infra-red detectors which are biased from the V_{ref} voltage through 1M resistors. These detectors are so sensitive when used on the analogue to digital converter input that they only need to be illuminated by an ordinary 60 watt lamp, 3 or 4 feet above the track. You could get the equivalent illumination by having a couple of torch bulbs on the underside of a bridge over the track. If suitable lamps are used they can be powered from the transformer used to power the cars.

Program

The facilities offered by this program are:

1) A slightly more interesting starting sound than the single track program.

2) A false start check to see whether either of the cars goes across the start/finish line before the third bleep of the starting sound.

3) A continuous readout of the number of laps completed by each car.

4) At the end of the race, it indicates who has won or whether there has been a dead heat.

5) Also at the end of the race, it produces a readout of all the lap speeds and average speeds at each stage.

The core of the program (**Listing 1**) is given in lines 40 to 220. After the initial setting up of the screen and various variables, we have a loop which consists of starting the race and then repeatedly looking at the detectors using the ADVAL command until both cars have finished the race, as signalled by the fact that both A% and B% are true (line 160). The results are then printed out and the user is asked if he wants another race.

Various techniques have been used to speed up the checking of the detectors. Firstly the ADC's are set in the 8 bit mode (line 1120) which means that they only take 4 milliseconds per conversion instead of 10 milliseconds. Secondly, integer variables (generally the resident integer variables A% to Z%) are used in all the time-critical parts of the program. Finally the most critical parts are put into multi-statement lines. In fact the program could be speeded up a bit more by putting lines 90 to 120 into one single line. However the program seems to work perfectly well as it stands, and it keeps the Art Editor happy if I don't use too many multi-statement lines!

The repeated "ADCAL1DIVJ%" or "ADVAL2DIVJ%" functions are used to pick off the top eight bits of the ADVAL value. (J% is a constant, equal to 256.) In the 8-bit mode, all of the top 10 bits of the 16-bit number returned by the ADVAL function change in response to the input voltage, and although the bottom four bits are zero as in the 12-bit mode, the 11th and 12th bits are permanently at logic 1. This means that when the detectors are fully illuminated, the value of ADVAL is 48 and not zero as it would be if we were using the slower 12 bit-mode.

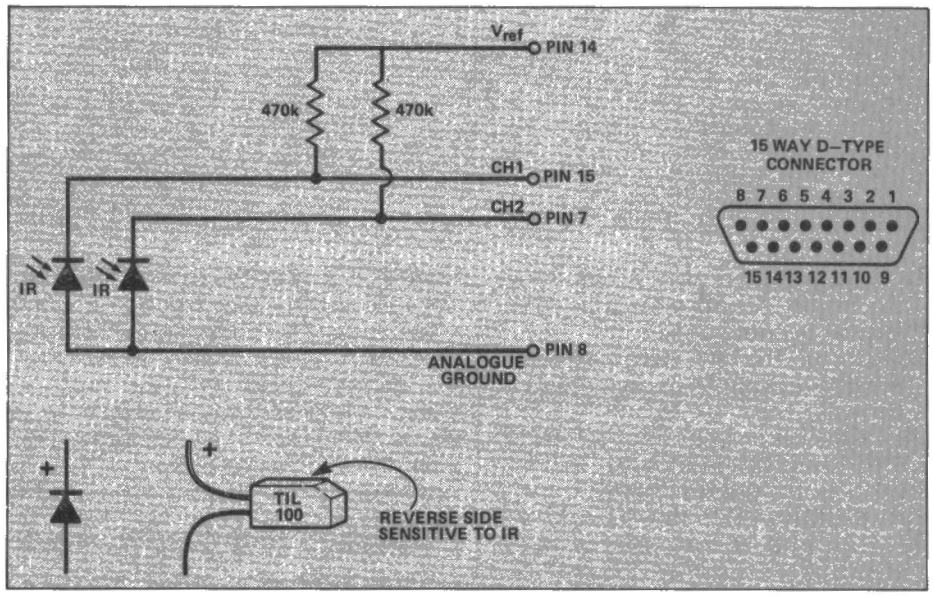


Figure 1. Circuit diagram of the infra-red detectors.

In-depth Analysis

In order to explain some of the techniques used, here are some detailed comments about the program by line number:

LISTING 1. The core of the program is in lines 40-220.

```
  10 REM Two track timer in BASIC.
  20 REM (C) November 1984
  30 REM Norwich Computer Services
  40 MODE 3
  50 PROCinitialise
  60 REPEAT
  70   PROCstart_race
  80   TIME=0
  90   REPEAT
 100     REPEATUNTILADVAL1DIVJ%ORADVAL2DIVJ%
 110     T%=TIME
 120     IFADVAL1DIVJ%P%=1:REPEATUNTILADVAL0DIVJ%=2
ELSEREPEATUNTILADVAL0DIVJ%=1:IFADVAL1DIVJ%P%=1
 130     IFADVAL2DIVJ%Q%=1
 140     IFT%P%=0:PROCa
 150     IFT%Q%=0:PROCb
 160     UNTIL A% AND B%
 170   PROCprint_results
 180   PRINT "Press space for another race."
 190   G=GET
 200   UNTIL G<>32
 210 *FX214,5
 220 END
 230
 240 DEF PROCa
 250 IF T%<R% ENDPROC
 260 IF A% ENDPROC
 270 L%=L%+1
 280 TA%(L%)=T%
 290 IF L%=laps% A%=1:VDU7
 300 R%=T%+200
 310 PRINT TAB(6,2);L%
 320 ENDPROC
 330
 340 DEF PROCb
 350 IF T%<S% ENDPROC
 360 IF B% ENDPROC
 370 M%=M%+1
 380 TB%(M%)=T%
 390 IF M%=laps% B%=1:VDU7
 400 S%=T%+200
 410 PRINT TAB(14,2);M%
 420 ENDPROC
 430
 440 DEF PROCstart_race
 450 CLS
 460 INPUT"Number of laps";laps%
 470 PRINT"      A       B"
 480 PRINT"LAP:"
 490 A%=0:B%=0
 500 P%=0:Q%=0
 510 R%=300:S%=300
 520 L%=0:M%=0
 530 *FX214,5
 540 REPEAT
 550   *FX213,200
 560   startA=TRUE:startB=TRUE
 570   FOR N%=1 TO 3
 580     FOR WAIT=1TO400
 590       IF ADVAL1DIVJ% startA=FALSE
 600       IF ADVAL2DIVJ% startB=FALSE
 610       NEXT
 620     VDU7
 630     NEXT
 640   VDU7
 650   IF (startA AND startB)=FALSE PROCfalse_start
 660   UNTIL (startA AND startB)=TRUE
 670 *FX213,100
 680 *FX214,20
 690 ENDPROC
 700
 710 DEF PROCfalse_start
 720 *FX15
 730 *FX213,10
 740 VDU7,7,7,7,7
 750 IF startA=FALSE PRINT'"False start A"
 760 IF startB=FALSE PRINT'"False start B"
 770 L%=0:M%=0
 780 PRINT'"Press space when ready"
 790 REPEAT UNTIL GET=32
 800 FOR S=4TO8STEP2
 810   PRINT TAB(0,S);SPC(24)
 820   NEXT
 830 PRINTTAB(0,2)
 840 ENDPROC
 850
 860 DEF PROCprint_results
 870 PRINT CHR$(12);
 875 PRINT "Lap Number  Lap speed A   Lap speed B";
 880 PRINT "   Average speed A   Average speed B"
 890 FOR lap%=1TOlaps%
 900   PRINT"    ";lap%;
 910   @%=&2010F
 920   PRINT track_length/(TA%(lap%)-TA%(lap%-1))*72;
 930   PRINT track_length/(TB%(lap%)-TB%(lap%-1))*72;
 940   ave_spdA = track_length*lap%/TA%(lap%)*72
 950   PRINTave_spdA;
 960   ave_spdB = track_length*lap%/TB%(lap%)*72
 970   PRINTave_spdB
 980   @%=10
 990   NEXT
1000 IF ave_spdA = ave_spdB PRINT '"DEAD HEAT":ENDPROC
1010 IF ave_spdA>ave_spdB PRINT '"A wins!!!":ENDPROC
1020 PRINT '"B wins!!!"
1030 ENDPROC
1040
1050 DEF PROCinitialise
1060 DIM TA%(100),TB%(100)
1070 VDU19;4;0;
1080 @%=10
1090 J%=256
1100 track_length=4*(34.9+25.6+32.3)+2*35.8
1110 *FX16,2
1120 *FX190,8
1130 ENDPROC
```

100 – Wait until either of the ADVAL values is non-zero. ie one or other of the detectors has been obscured.

110 – Capture the TIME value even before trying to work out which car it is that has arrived.

120 – If the car on track 1 has arrived set P% = 1 and wait until the ADC returns a value for the second channel. If not, it must have been the other track, so wait until the ADC returns a value for track 1 and if that one has arrived too, set P% = 1.

130 – If car 2 has arrived, set Q% = 1.

140 – If P% is 1, zero it again and deal with the fact that car 1 is over its detector.

150 – Ditto for Q% and the second car.

PROCa and PROCb have a similar structure, so we will just look at the first one. During the time that the car is moving over the detector, the procedure will be entered repeatedly since the detector is obscured and the ADVAL value is not zero variable. The R% (or S% for car 2) is used to indicate whether a reasonable time has elapsed since the last time that the beam was cut. While the car is still obscuring the beam the program must not register another lap. Therefore, the fact that the beam is obscured is ignored until enough time has elapsed for the car to have come round the track again. In the first instance R% is set to 300 (line 510) so that until 3 seconds have elapsed from the start, any cutting of the beam is ignored.

250 – If the time has not gone far enough beyond the previous reading, give up.

260 – If car 1 has finished all its laps, give up.

270 – Otherwise increase the lap-count.

280 – Record the time for the end of that lap.

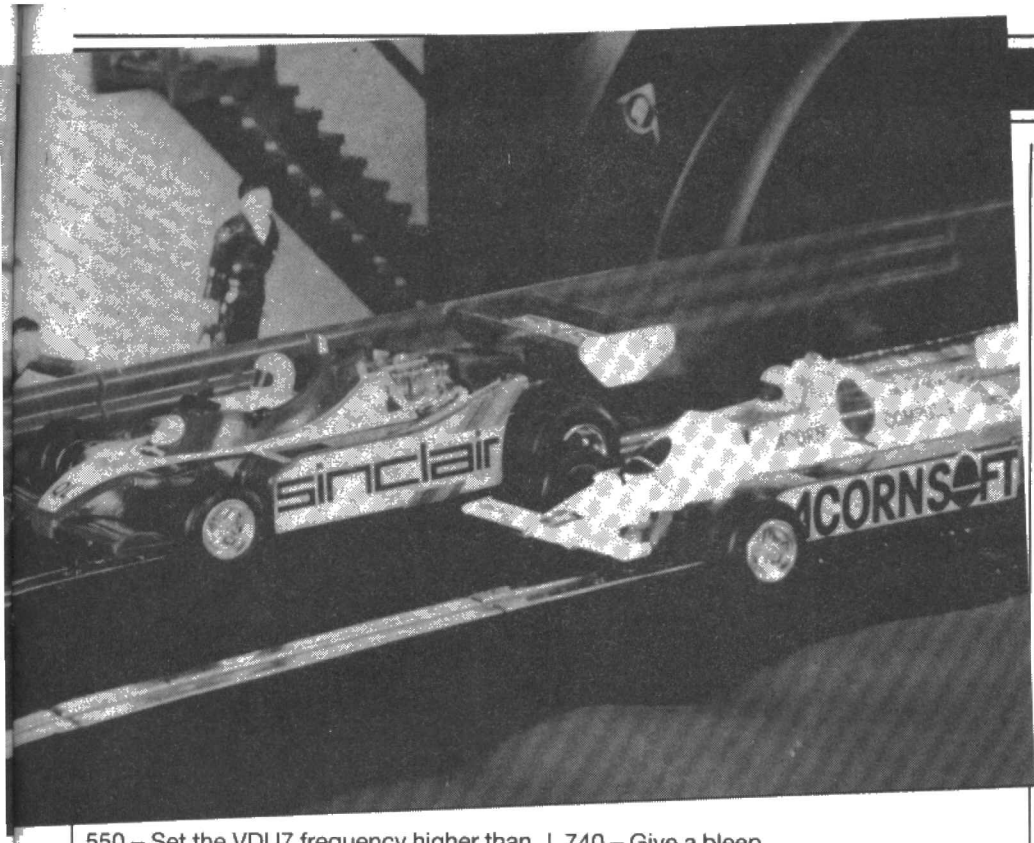
290 – If car 1 has done the full number of laps, signal the fact with the variable A% and give a bleep.

300 – Set the minimum time allowable before checking the ADVAL value again.

"Various techniques have been used to speed up A/D conversion . . . this avoids the use of machine code and interrupts".

310 – Print out the lap number.

At the start of the next race, the number of laps for the race is asked for and various of the variables are re-initialised (lines 460 to 530). Then we enter a loop which produces start bleeps and checks for false starts:



550 – Set the VDU7 frequency higher than normal.
560 – Initialise the logical variables that are used to indicate whether or not there has been a false start ie whether the starts are TRUE or FALSE.
570 – For three bleeps.
580 – Wait a bit.
590 – If beam 1 is obscured, car 1 has made a false start.
600 – Ditto for car 2.
610 – Make a bleep.
640 – Add an extra bleep to make the last bleep longer than the other two.
650 – If either car has made a false start, act accordingly.
660 – Only finish the starting procedure when both have made a clean start.
670 – Reset VDU7 frequency.
680 – Reset VDU7 duration.
If there has been a false start:
720 – Flush all buffers in order to stop the bleep.
730 – Give VDU7 a low pitch.
740 – Give a bleep.
750 – If car 1 did a false start, indicate on the screen.
760 – Ditto for car 2.
770 – Reset the lap counters to zero.

Pressing the space bar will re-enter to the start sequence, restoring the display ready to show lap counts again. The printing out of results in lines 860 to 990 is straightforward, as is the declaration of the winner at lines 1000 to 1020.

"There are a number of ways in which to expand the system . . . the addition of warning lights for example".

The various bits of initialisation are:
1060 – Save space to record up to 100 laps, though for this many laps you would need to put a printer onto the computer to record the results. There is only room on the screen in mode 3 for the results of 20 laps plus the declaration of the winner.
1070 – Change the background colour to blue.
1080 – Set the formatting at 10 character columns.
1090 – J%, which is used to divide the ADVAL values.
1100 – The track length in cm is set in terms of the number of pieces of each of various different lengths.
1110 – Only enable two ADC channels.
1120 – Switch to 8-bit mode.

Possible expansion

There are a number of ways in which the system could be extended. For example it would not be difficult to improve on the start sounds by using the SOUND command rather than VDU7. Then the program could be personalised by allowing the drivers to input their names at the beginning of each race so that it would tell you that Fred has won rather than just "A" or "B wins"! It would be a nice touch, when inputting names, to give prompts like, "What is the name of the driver on the stand side track?", or however you want to identify the two tracks. Another option could be to use the same names again for a repeat race to save retyping them.

One idea for making things more interesting if you like playing with the hardware would be to make some warning lights. LEDs could be connected to the printer port to give visible warning to the drivers so that things look more realistic. The slot-car manufacturers do produce starting lights but these are extremely expensive. All that is needed is a printer port cable, some LEDs and some 100 ohm resistors. The printer port has a buffer chip on it which is easily able to supply 10 mA to any LED on each of its 8 channels. Different coloured LEDs could be used along with duplicate lights on each side of the track provided no more than 4 lights are required on each side.

Lighting up time

Programming the lights is simplicity itself. Give each LED the number value shown in **Figure 2,** then to light the third LED simply say ?&FE61 = 4. The address of the printer port is, &FE61 and, by putting out the number 4, a logic one is produced on the third line and zeroes on all the others. To light more than one LED at a time, simply add up the numbers, for example to switch on the first and third 'lights' use ?&FE61 = 5. That's really all there is to it. Obviously the project can be developed further according to readers' imagination and technical innovation.

Next month we will look at a way of improving the resolution of the timing by using the User Port and the 6522 VIA timers.

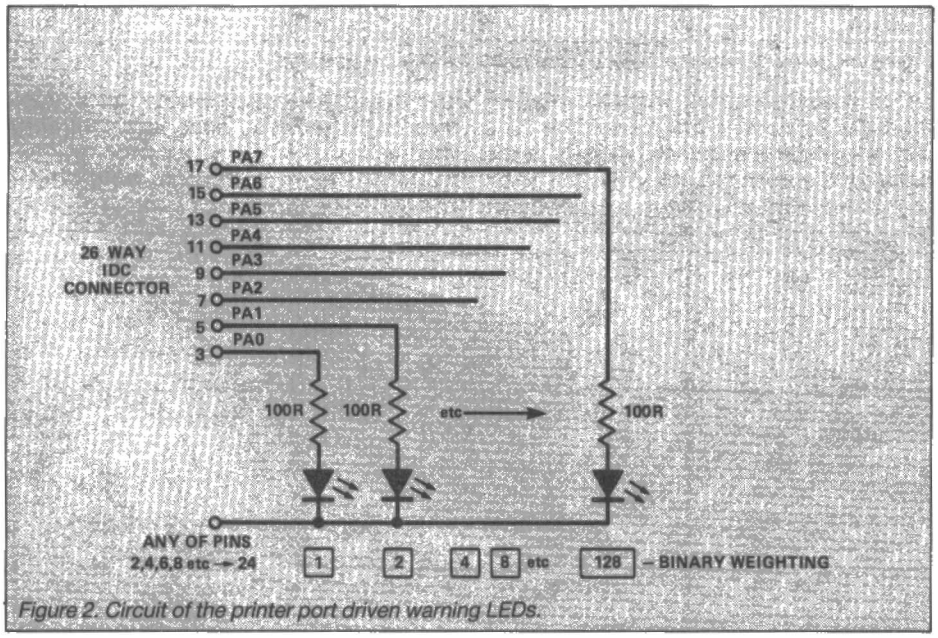


Figure 2. Circuit of the printer port driven warning LEDs.

RIBBON ECONOMISER

Save the frustration and expense of replacing printer ribbons with Dick Leslie's cartridge modification. Your printer ribbons will last and last and . . .

Picture, if you will, a familiar scene. Having spent all weekend on a particularly interesting program that just requires debugging, you hook up your printer and sit back as the sheets start to unfold. Suddenly, in front of your disbelieving eyes, the ink density of the dots starts to go a distinct shade of off-white and all the signs are that, despite a recent replacement, the ribbon once again needs changing. Recognise the scenario? This article presents an inexpensive solution to this frequently exasperating problem.

An endless loop of fabric inside a long, narrow plastic box forms the "Epson type" cartridge ribbon, and this is found (naturally) on all Epson printers and on a great many Epson look-alikes. Although the price of the cartridge has fallen drastically in recent years, it is still about £5 – £6 in one-off quantities, and it always seems to need replacing despite a modification which can be made to the cartridge which will prolong the ink density.

Inside the cartridge

The first point to make is that the cartridge is made to be dismantled, and that replacement ribbons can be bought and fitted inside. The replacement ribbon (usually only available on special order) costs about a pound and is, as such, good value for money. The fun comes when you start to fit this replacement but we'll come back to this problem shortly.

The cartridge case must be opened with care that is if the plastic retaining clips are not to be broken. There are usually six of these clips, and they can clearly be seen on the cartridge case. What cannot be seen are the ten plastic lugs on the inside of the cartridge lid which engage with ten holes in the cartridge box. Therefore, the lid must be gradually eased off without snapping the lugs. To do this, first loosen the clips with your thumb and then, starting at the right-hand end on the box, prise the lid off using gentle pressure from a screwdriver inserted between the lid and the box. Work your way around the lid, raising it a millimetre at a time until it finally comes off.

Inside you will see yards and yards of inked ribbon neatly and densely packed in tight loops within the box. A capstan and pressure wheel arrangement pushes the ribbon into one end of the box, while at the other end the ribbon is drawn out by the same wheels. A leaf-spring exerts pressure on the ribbon before it leaves the box, thus putting the section of ribbon which passes the dot-matrix needle-head under tension. When buying a replacement ribbon, the user has to thread it into the wheel-assembly and continue turning the capstan (by hand) until the entire ribbon is packed into the box. When you remember that the new ribbon is quite damp with fresh ink you will appreciate why computer store staff report that there is "little demand for replacement ribbons".

The modification to the ribbon cartridge works because the ribbons always dry out before they wear out. If you're in the habit of leaving your printer switched on all day, the ribbon will dry out even when you're not printing; it gets quite warm inside a dot-matrix printer. Refreshing dry ribbon with stamp-pad ink is the solution to the problem, but it must be done in a controlled manner.

There is room on the right-hand side of the ribbon cartridge for a swab of sponge,

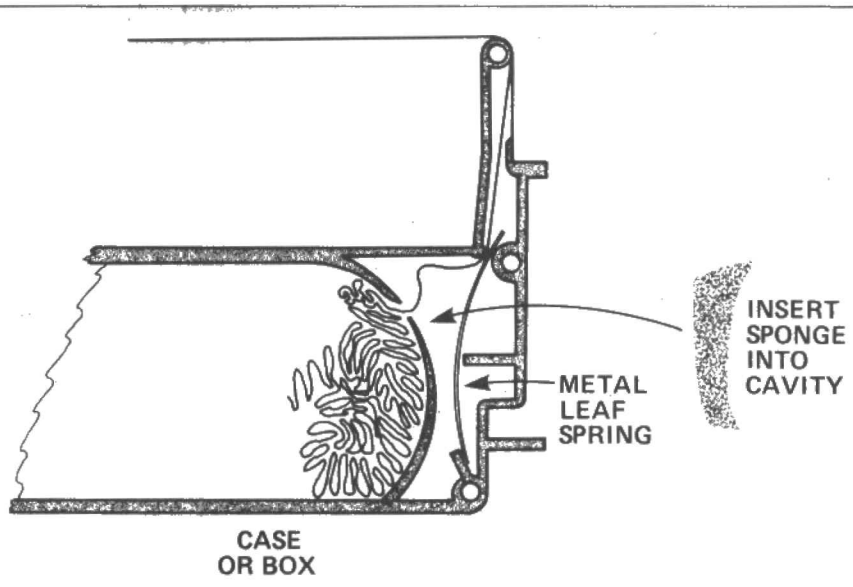


Figure 1. Use the spring to prevent excessive ink leaving the cartridge.

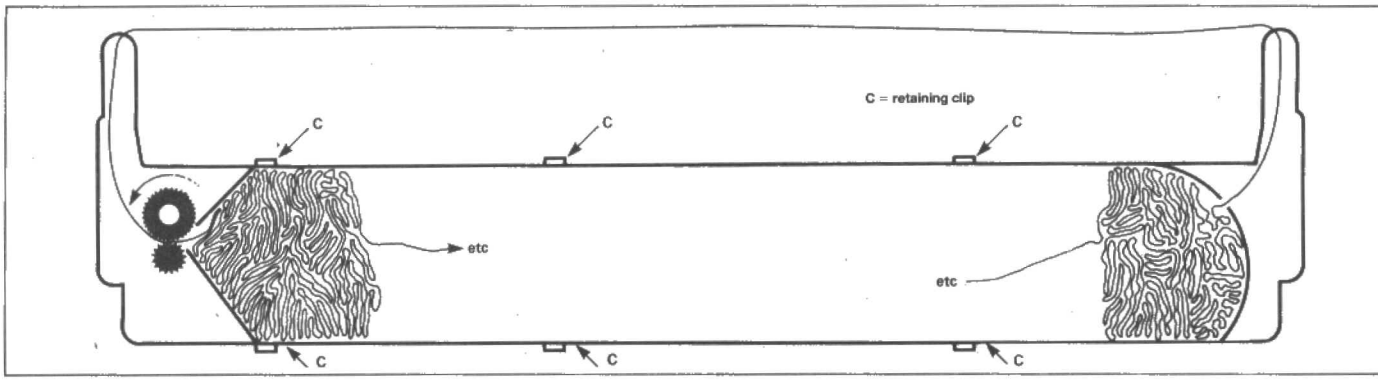


Figure 2. Care should be taken not to break the six retaining clips when opening the case.

which can be cut and positioned in such a way that the ribbon presses up against it on its way out of the box. A sponge soaked in stamp-pad ink will keep the ribbon well-inked until it physically deteriorates which takes quite a long time.

"Make this modification and your ribbons will seemingly go on for ever".

Refill bottles of stamp-pad ink can be bought from any stationers or printers, and you should not use any other type of ink since it may not have the right properties. Epson ribbon cartridges do not have seams on their lower edges and will not leak ink. The box-structure of the cartridge case forms an ideal reservoir, but on no account should it be used as such! The idea is to keep the sponge damp, not to form an ink-well. The metal leaf spring will prevent excessive ink leaving the cartridge, but if you do over-ink the sponge, just wind the ribbon by hand for a few minutes to clear the over-inked section away. In general, over-inking is not a problem, because two holes will be drilled in the cartridge lid, and a regular inking routine will be established.

The larger hole, some 6mm in diameter, is drilled in the position shown on the diagram. Through it you will be able to see the edge of the sponge where it touches the ribbon, and you will be able to judge how damp the sponge is. A second hole some distance away from the first is useful as a filling hole. The sponge furthest from the ribbon receives the most ink, which gradually moves by capillary action to the business end of the sponge. Of course, there is nothing to stop you putting ink through the viewing hole if you want quick results! An old eye-dropper is ideal for feeding small quantities of ink to the sponge.

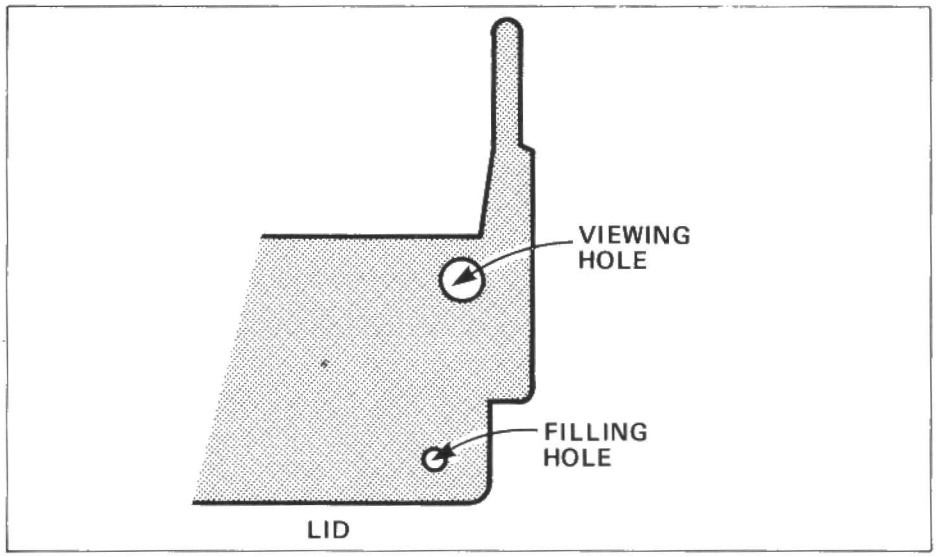


Figure 3. Drill two holes in the cartridge as shown.

Finally, you may find it worthwhile spreading drops of ink over the entire coiled-up ribbon before you put the cartridge lid back on. The sponge will take many days to revitalise a dry ribbon if it is not given this help.

When in use it is better to give the sponge a drop of ink per session than to forget about it and then give it a large helping of ink just prior to a print run. Your ribbons will seemingly go on forever and you'll always have readable print-outs.

FREE EVERY MONTH

Your ROBOT

BRITAIN'S FIRST ROBOTICS MAGAZINE

Each month, Your Robot reports on the latest news from the world of low cost robotics. In addition in-depth features explain the theory behind the operation of robots and computer control systems.

HI-TECH ROBOTS AND EDUCATION

*News this month concentrates on the **Hi-technology and computers in education** exhibition that took place at the Barbican between January 23rd and 26th. Many of the major suppliers of robotic equipment to home and education users were present including **Commotion** who, in addition to their range of robotic devices including the new infra-red controlled Beasty servo system, were selling their new 32 page catalogue for 50p. This publication is a must for anyone interested in experimenting with low-cost robotic systems. **Colne Robotics** had their well established Armdroid desk top arm on display while **Remcon** were demonstrating their newly released Teach Robot that managed to perform faultlessly for the full duration of the show. The construction of this arm will shortly be described in **Your Robot,** plans to publish this article were delayed due to production problems at the **Your Robot** offices but we shall be going ahead in the April issue.*

*The ogre arm from **L. Staines** was also on display and generated considerable interest as did the Mentor and Neptune robots from **Cybernetic Applications.** This latter robot is a powerful hydraulic arm that avoids using the traditional (messy) fluid but makes use of ordinary tap water and thus avoids the problems often associated with this means of driving an arm.*

The show itself was voted a success by those that attended and a similar event is planned for early next year.

*We apologise to readers of **Your Robot** for the curtailed coverage of this aspect of computing during recent months and in particular to the lack of space devoted to the magazine this month. We are examining ways of extending the content of **Your Robot** and expanding the range of material later during the year.*

YOUR ROBOT NEWS DESK 01 251 6222

WIN A BRAND NEW MANCOMP MO184 PROFESSIONAL KEYBOARD FOR YOUR ZX SPECTRUM

All you have to do is list the features below in the order that you consider to be most important when selecting a keyboard for your ZX Spectrum. For example, if you consider 'Punctuation' to be most important then put 'A' in the appropriate box and so on until you have listed all the features below.

- ☐ NUMERIC PAD (0-9 and Decimal Point)
- ☐ MATHEMATICAL NOTATION (+, −, =, < >,*)
- ☐ PUNCTUATION (Full Stop, ; : * $)
- ☐ FACILITY TO HOUSE POWER SUPPLY INSIDE KEYBOARD
- ☐ CURSOR KEYS
- ☐ SAME LAYOUT AS SPECTRUM+
- ☐ FACILITY TO HOUSE INTERFACE 1 INSIDE KEYBOARD
- ☐ HANDREST
- ☐ KEYS LEGIBLE AT THREE FEET

As a tie-breaker we would like your own suggestion (in not more than 15 words) as to a feature you would like to see on a keyboard. (BLOCK CAPITALS PLEASE)

_____

_____

FILL IN THE FORM BELOW AND SEND IT TO:

***E&CM* Keyboard Competition, Priory Court, 30-32 Farringdon Lane, London EC1R 3AU**

NAME _____
(BLOCK CAPITALS PLEASE)

ADDRESS _____

_____

_____

Please note the Editor's decision will be final on all aspects of the competition. No EMAP employee or members of their family are eligible for entry.

LOW COST ROBOT SYSTEMS

CLOSED LOOP CONTROL SYSTEMS

Richard Sargent looks at closed loop control systems – a vital part of many robot designs.

As a robot arm moves about its working envelope it can easily lose track of where it's meant to be. Whereas a floor-buggy can work perfectly well with little or no positional *feedback,* an arm can do nothing useful if it can't relate to the object it's working with. If you place an ordinary DC motor, the LEGO motor for example, on a polar arm, you will have an OPEN-LOOP situation where the motor drives the arm blindly. A computer program may tell the arm to "raise jib 2 seconds, swivel base clockwise 1 second", but there is no guarantee that the instructions will bring the gripper to its intended destination. At first sight, such an open-loop arrangement doesn't seem worthy of consideration. However, replace the DC motor with a stepper motor and even open-loop arrangements can be quite accurate. The commands "raise jib 20 steps, swivel clockwise 10 steps" may well produce the same accurate result at any room temperature and with any reasonable weight of gripper load. The stepper motor moves in set increments, and, in theory at least, is not subject to slip or over-run. Providing you transmit its power by chain and axle (rather than by pulleys or gearing) then the steps sent out by the computer should faithfully be reproduced as arm movements. The main advantage of the open-loop system is that it is cheap to implement.

Closed loop systems, on the other hand, are always extremely accurate. Here, the computer constantly monitors the position of the arm and will continue to drive the motors until the position specified has been reached. In electrical terms, the desired position must match the actual position, and until this is so an "error signal" drives the motor in the appropriate direction. (See **Figure 1).**

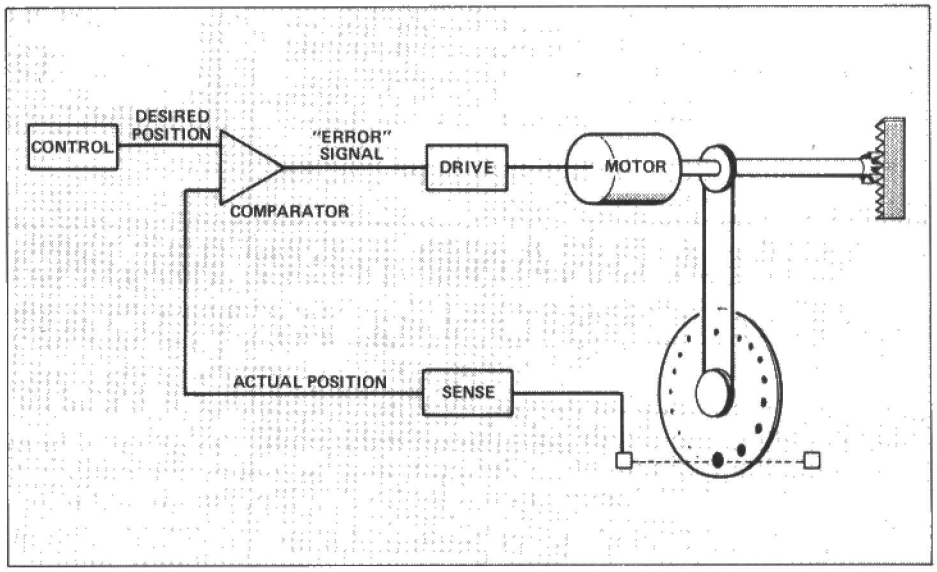
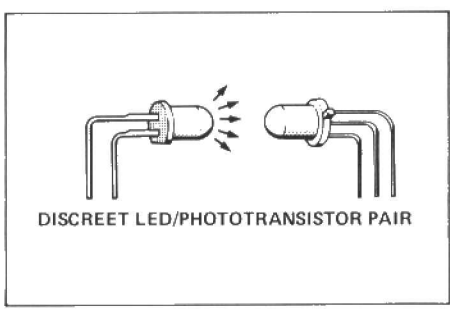
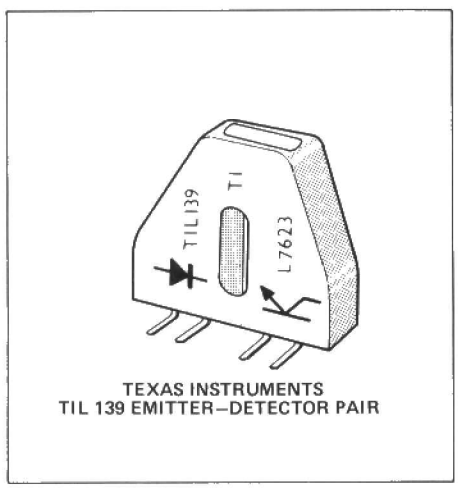


Figure 1. Block diagram of typical closed loop systems.

There are two common methods of measuring mechanical movement, and each method has its preferred type of sensor. The first way is to measure rotary movement. From the figures obtained both motor speed and distance moved can be calculated. This method has been around a long time. On a car the device is the speedometer/mileometer combination and the sensing device is magnetic. For low-cost robotics the optical sensors shown in **Figure 2** are a sound choice. In





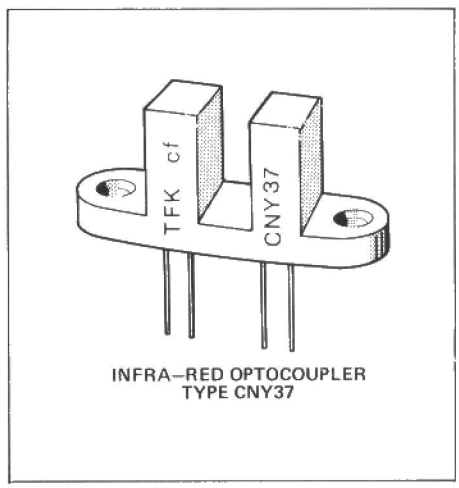


Figure 2. Physical outlines of a range of popular optical sensor devices.

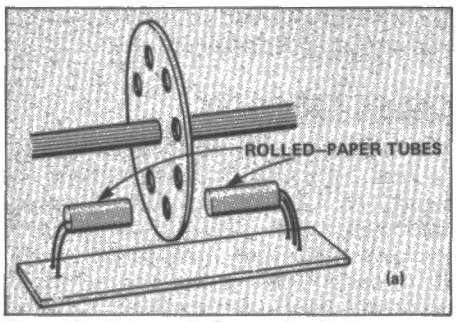


Figure 3a. Robust shaft encoder design.

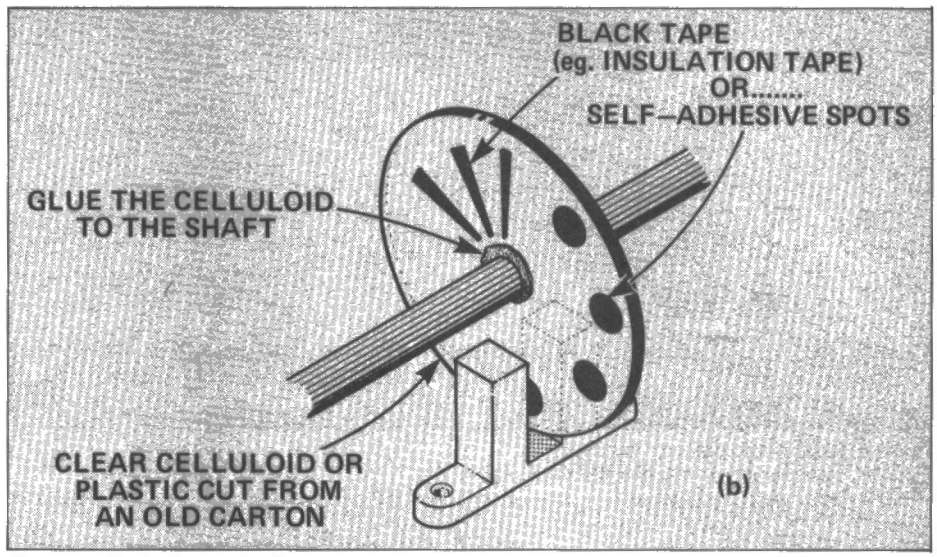
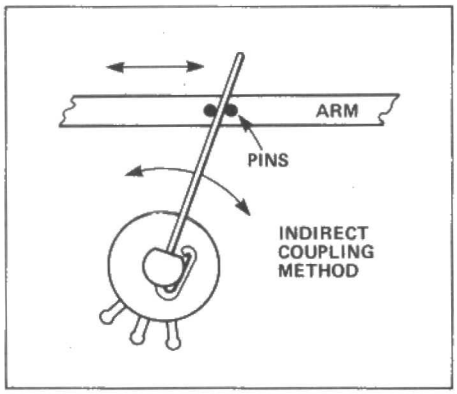


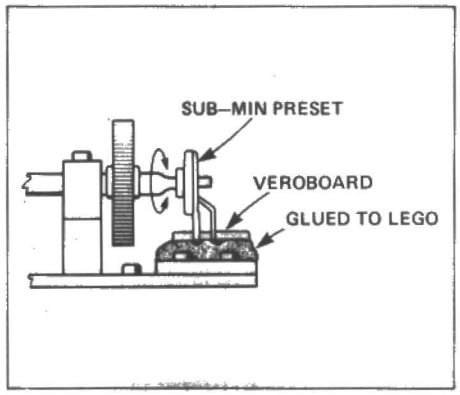
Figure 3b. Typical encoder disk design.

all cases the beam of light is intermittently broken and provides a train of square-wave pulses which can easily be counted by the computer. If the sensor is liable to rough handling then the perforated-gear-wheel option, **Figure 3a,** is obviously a better choice than the spots-on-celluloid option of **Figure 3b.**

As a general rule-of-thumb you'll need just one beam-interruption per revolution if you intend to put the disc directly on the motor drive-shaft, and as many interruptions per revolution as you can manage if you intend to put the disc on a drive wheel or arm joint. The electronic circuitry of this opto-sensor will be examined shortly.

The second method of measuring movement is known as discrete-position sensing. In its simplest form it is represented by the bump-sensors of Hardy of the limit-switches of Wall-Builder. However, a robot arm requires continuous position sensing, and there are two options available which will provide this. The one which you use will rather depend on the facilities offered by your micro. The preferred option is to fix a linear carbon-track potentiometer to each joint of the robot arm. (See **Figures 4a, 4b**). As the arm moves, so the potentiometer's shaft moves the wiper over the carbon track, generating a continuous change in resistance. The output from the potentiometer is a changing DC voltage which can be monitored by the analogue input of your micro. Software then calculates the actual angle taken up by the arm.





Figures 4a and 4b. Two methods of mounting pots.

"Closed loop systems allow accurate control".

Note that since most potentiometers can only turn through 300° at best, this method might not be suitable for some joints, particularly the wrist joint where 360° rotation is useful. And what if you have a Spectrum rather than a BBC micro? There's no analogue port available for Spectrum users so a method which unfortunately has far less resolution than the potentiometer has to be used.

ALTERNATIVE APPROACH

Figure 5 shows the method in question. Light sensors are used in conjunction with a specially coded disc to produce a TTL Bit-pattern which represents the *approximate* angle of the joint. The light source can be a filament bulb running on a smoothed DC voltage rather than three separate LEDs as shown in the diagram, but if you opt for this light source be sure to use photodiodes which respond to natural light rather than those which respond to infra-red light.

The coded disc is an interesting device in its own right. It uses a special code known as the "Gray Code" in which only one bit ever changes as the count proceeds from one state to another. Consider the phototransistor taking a "read" on the boundary between the third and fourth sectors of the disc. With the Gray scale the result will be "3" or "4" – the sensor might be confused but it's not going to be far wrong. On the other hand the traditional binary-count would lead to spurious readings caused by *more than one bit* changing as the count progressed – the phototransistor would invariably "catch" the changed-state of one bit before any other and so the boundary between "3"(011) and

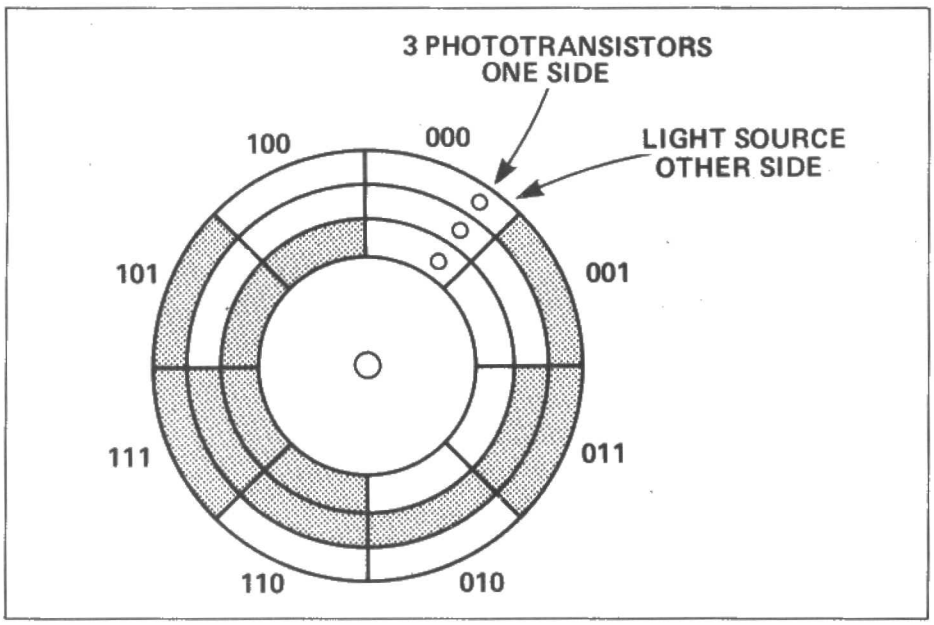


Figure 5. A gray code encoded disk.

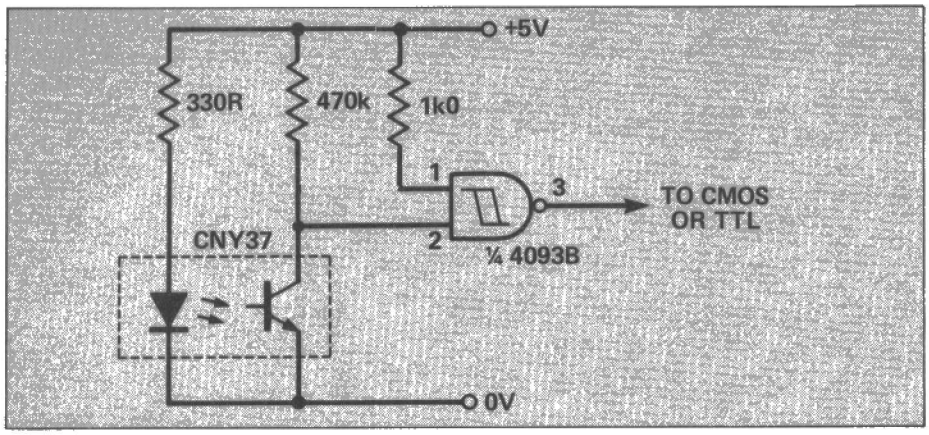


Figure 6. Basic photo detector interface.

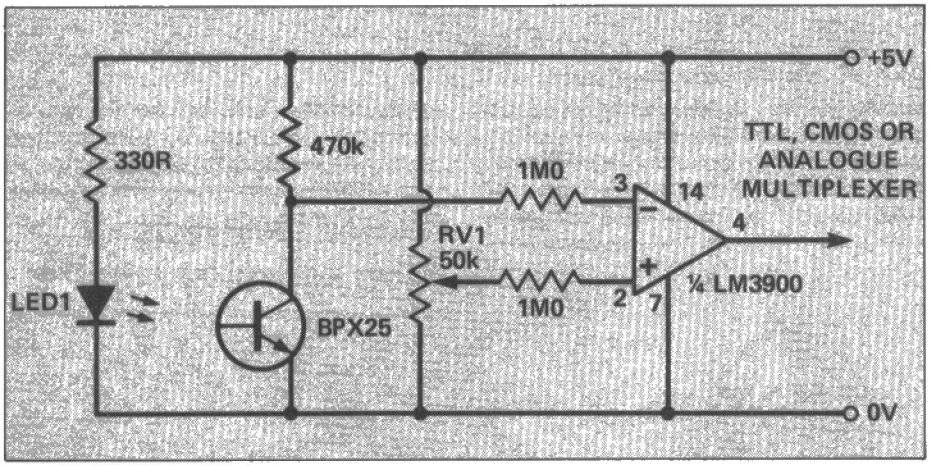


Figure 7. Detector interface with variable sensitivity.

"4"(100) might be read as 110 or "6" which wouldn't be acceptable!

SENSE CIRCUITS

The output from the phototransistor may be read by a CMOS Schmitt trigger as shown in **Figure 6.** The infra-red optocoupler unit CNY37 doesn't need shielding from room lighting and is well worth using. There is little "noise" introduced into optics, and the Schmitt trigger is capable of producing a very clean waveform. However if you are using ordinary light/phototransistor combinations you might find that the circuit featured in **Figure 7** is worth building. Under ambient lighting levels the waveform at the collector of the phototransistor is ragged and rides on some DC offset level. Using the Op-Amp with the sensitivity control, VR1, enables the unwanted light to be forgotten while at the same time applying the true signal to useable levels. The phototransistor switches OFF when the light is blocked and the output of the amplifier goes close to zero.

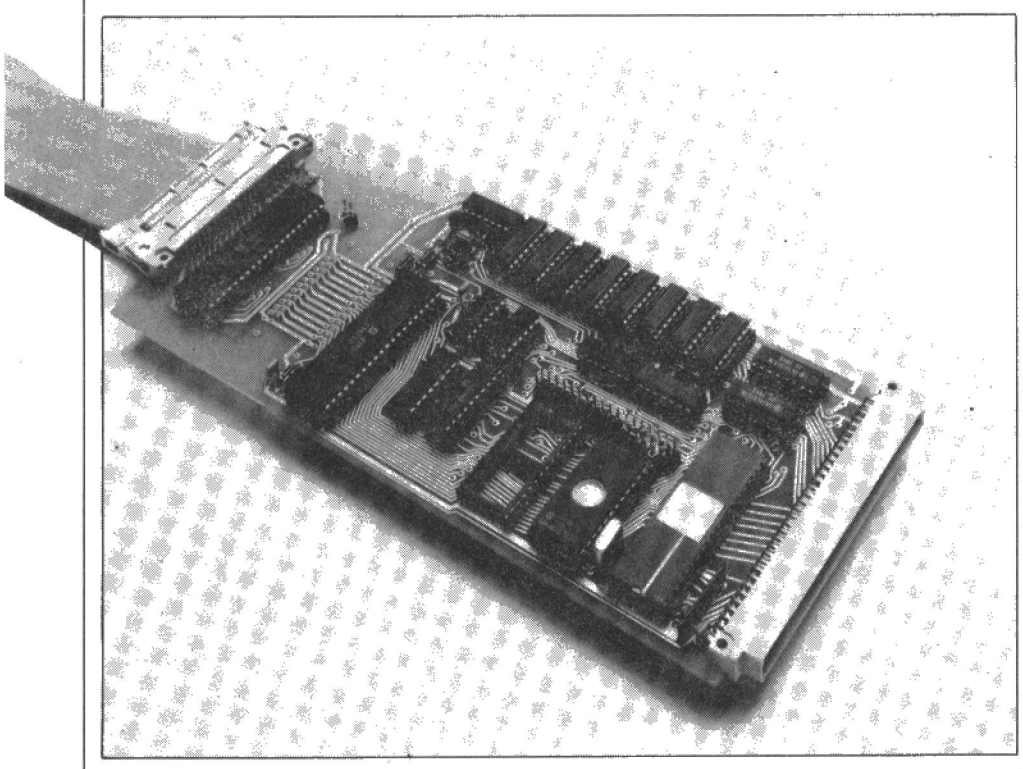
"The coded disk is an interesting device".

Next month we shall look at closed-loop techniques using potentiometers, with circuits and software to drive a 5 axis arm.

6809 PROCESSOR

Mike James reviews the CMS 6809 second processor card that allows Beeb owners to explore the power of the FLEA operating system.

These days it is becoming increasingly difficult not to think of the BBC Micro as just a host for one of the many second processors and other peripherals that are available. Many critics have commented that as a 'stand alone' computer the Beeb is beginning to show its age and, to a certain extent, this is true but as the core of a computer system it still looks youthful and vigorous. By adding a second processor of one kind or another the Beeb's main problem, lack of user RAM, can be overcome. In addition access may be gained to a whole new range of software or old 6502 software



run much faster. Although *E&CM* did a review of second processors in the November issue one interesting item slipped through the net – the CMS (Cambridge Microprocessor Systems) 6809 card. Although the main reasons why this card is of interest are contained in this review briefly:

1) it gives BBC Micro users a low cost route to the excellent 6809 FLEX operating system and the entire range of FLEX software.

2) it gives FLEX users access to the BBC Micro's graphics and the full range of its peripherals.

3) it fits inside the BBC Micro's case thus avoiding adding yet another box to the system.

Simplicity itself

Essentially a computer is a processor, some RAM and a little ROM! Of course as soon as the inevitable interfaces to the outside world are added things are a lot more complicated. This is where the BBC Micro comes into the picture – it has all the interfaces that any computer could want. This means that, in principle, designing a second processor should be simply a matter of putting together a processor with some RAM/ROM and solving the problem of interfacing to the tube. (See "Battle of the Second Processors" in the March issue of *E&CM* for technical details of second processor design.) Some designers have found this final step a little daunting and have used one of the alternative BBC interfaes, the user port, the 1MHz bus and even the RS423 port! However CMS have managed to recognise the simplicity of using the tube and as a result their 6809 card is elegant and powerful.

Back to back

Instead of developing a custom chip as Acorn have done, CMS have used the method of connecting a pair of 6522 VIAs back to back. That is, one VIA is interfaced to the 6809 system and its data lines are connected to a second VIA interfaced to the BBC Micro via the tube. Passing data betwen the two systems is as simple as storing data in a memory location and it works almost as quickly. The VIAs even provide the extra handshaking lines needed to co-ordinate the transfer. Both VIAs are mounted on the 6809 card along with a 6809, eight 64K DRAMS and a few TTL chips. An EPROM containing the monitor software completes the 6809 hardware and confirms that adding a second processor is mainly a matter of software.

The RAM and ROM are configured as standard full 6809 FLEX memory maps, that is 56K of RAM (from 0 to C000) with the remaining 8K divided between a 4K monitor ROM, some user RAM/ROM and system I/O. Of course, in this case, system I/O comes down to a single VIA so any FLEX software that expects to find an ACIA at E000 etc will be very unhappy. This lack of a range of I/O devices in the memory map may sound like a serious problem but it isn't for two reasons – firstly all good (ie nearly all!) FLEX software uses FLEX system calls to do its I/O and secondly all of these routines have been redirected to use I/O facilities offered by the BBC Micro on the other side of the tube.

Fitting and use

The CMS 6809 card is very easy to fit to the BBC Micro – take the cover off, slip the interface cable under the keyboard and plug in to the tube; finally stick (!) the 6809 PCB to the underside of the cover and replace it. Once installed the 6809 card is hidden from view and adds nothing to the space that your BBC Micro occupies.

Getting the system to run FLEX is just as easy. First insert a BBC Micro disk containing the communication software that runs

on the BBC side of the tube into one of the drives and load it. The communication software is a small machine code program that handles the transfer of bytes over the VIA link with the 6809 card. For most of the time it is simply passing ASCII data, a byte at a time from the keyboard and to the screen. Even this simple activity provides more computing power than you might expect. By making a connection between the BBC and FLEX's I/O drivers most of the facilities of the BBC Micro become available to FLEX software. For example, VDU control codes can be used to configure and drive the text and graphics screen; the BBC Micro's screen editing keys all work, you can use CTRL/B and CTRL/C to start and stop the printer etc . . . Add a few more routines to transfer blocks of data, pass OSBYTE and OSWORD calls and the software part of the connection is complete!

At first the only software running on the 6809's side of the tube is the CMS monitor. This is a fairly standard monitor with documented machine code subroutines for the assembly language programmer. In most cases the monitor is quickly replaced by FLEX by placing a system disk in drive 0 and typing "U". From this point on the BBC Micro and its disk drives behave like a standard (but sophisticated!) FLEX system.

If you know FLEX and its associated software there is little more to say. I have swapped 40 track disks between the BBC/FLEX and other FLEX systems with no trouble. All of the FLEX software that I have tried worked without modification. In fact because of the fast response of the screen and the editing keys most of the software proved easier and more pleasant to use.

Applications

There are a number of reasons why it is desirable to add a 6809 second processor to a BBC Micro. The most obvious is that like me you value FLEX and its associated software and just want a modern machine to run it. This of course brings with it the advantages of BBC graphics and sound both of which can be used from FLEX BASIC, C, PL9, Pascal, Assembler and Cobol! Of course all of these languages (apart from Assembler) have to be purchased separately but at least they are available. There is also a lot of high quality commercial software available but apart from configurable word processors it is unlikely to make any use of the extra graphics facilities offered by the BBC Micro.

There is a second and less obvious reason for wanting to add a 6809 to the BBC Micro – software development. CMS have a single board 6809 that is identical to the second processor card but minus the second VIA. This means that 6809 software can be developed using the BBC Micro and then transferred to a stand alone card once it is ready. In addition a wide range of other cards can be added to either 6809 processor via an expansion bus connector. The current range includes, a 512 x 256 four colour video display, a 12-bit 8 channel A to D, an IEEE interface, a serial/parallel interface, an 80 line parallel interface and an industrial controller capable of switching up to 16 channels with a maximum rating of 240V. Once you realise that all these extras are available it is difficult not to see the 6809 card as a way of providing the BBC Micro with a range of I/O interfaces rather than the other way round! More seriously if you are involved in the application of micros to control then a BBC Micro provides an excellent host for a full 6809 development system.

"There is a second and less obvious reason for wanting to add a 6809 card to the BBC – software development".

The 6809 card with tube interface and linking software (£274 plus VAT) and the other hardware/software mentioned in this article can be obtained from Cambridge Microprocessor Systems, 44a Hobson Street, Cambridge CB1 1NL. Telephone: 0223 324141.

COMMS COLUMN

This month Ben Knox gives PSS details together with comprehensive Bulletin Board information.

Packet switching systems

After January's Communications column covering PSS, I have had a number of requests to list all the PSS exchange numbers. So here goes:

Town	300 Baud	1200/75 Baud	1200/1200 Baud
Aberdeen	0224 642242	0224 642484	0224 642644
Belfast	0232 8281	0232 8291	0232 8201
Birmingham	021 214 5139	021 214 6191	021 214 3061
Brighton	0273 851111	0232 852111	0273 853111
Bristol	0272 216411	0272 216511	0272 216611
Cambridge	0223 82511	0223 82411	0223 82111
Cardiff	0222 376111	0222 376171	0223 376191
Edinburgh	031 337 9141	031 337 9121	031 337 9393
Glasgow	041 204 2011	041 204 2031	041 204 2051
Ipswich	0473 671111	0473 672111	0473 673111
Leeds	0532 470711	0532 470611	0532 470811
Liverpool	051 211 0000	051 212 5127	051 213 6327
London	01 825 9421	01 407 8344	01 928 2333
London	01 928 9111	01 928 3399	01 928 1737
London	01 840 0688	01 840 1399	01 840 5500
Luton	0582 8181	0582 8191	0582 8101
Maidstone	0622 88511	0622 88611	0622 88711
Manchester	061 833 0242	061 833 0091	061 833 0631
Newcastle	0632 314171	0632 314181	0632 314161
Nottingham	0602 881311	0602 881411	0602 881511
Portsmouth	0705 53011	0705 53911	0705 53811
Reading	0734 389111	0734 380111	0734 384111
Sheffield	0742 414171	0742 414181	0742 414191
Slough	0753 6141	0753 6131	0753 6171

To access PCC, dial the number of your local PSS exchange. When you hear the carrier tone, connect your modem to the telephone line. Wait for a few seconds to allow the carrier signals to lock. Type <RETURN> <RETURN> (the return key twice!) followed by D1, followed by another <RETURN>. Repeat this sequence if nothing appears on your screen. After receiving the PSS identification, enter your NUI, remembering to type N first. Next, type in the NUA of the computer you want to access and you will be connected.

BABBS 1 – Felixstowe (0394) 276306
- SYSOP: Tony Game
- TIMES: 24 hours
- SPEED: 300 Baud
- SIGS: Adventure Clues, CP/M, DOS, Hardware, Mac, Modem Spot, Sales and Wants
- NOTES: Download and Upload area, British Apple System User Group's BBS

BABBS 2 – (0268) 776956
- SYSOP: Mike Jones
- TIMES: 24 hours
- SPEED: 300 Baud
- SIGS: Adventure Clues, CP/M, DOS & PRO-DOS, Hardware, Insults and Graffiti, Jokes, Macintosh, Modems, Pascal, Sales & Wants
- NOTES: Upload and Download, British Apple System User Group's BBS

CABB – London (01) 631 3076
- SYSOP: Tony Dennis
- TIMES: 24 hours
- SPEED: 300 Baud; 300 and 1200/75 Baud Weekdays
- SIGS: Acorn, Commodore, CP/M, Lonely Hearts, Sinclair
- NOTES: Computer Answers Magazine BBS, program download area

CBBS London – (01) 399 2136
- SYSOP: Peter Goldman
- TIMES: Sundays only, 5.00p.m. – 10.00p.m.
- SPEED: 300 Baud
- NOTES: Otherwise known as 'MG-NET'

CBBS Surrey (04862) 25174
- SYSOP: Mike Parker
- TIMES: 24 hours
- SPEED: 300 Baud

CBBS SW (0626) 890014
- SYSOP: Boyd Hitchcock
- TIMES: 24 hours
- SPEED: 300 and 1200/75 Baud

City BB – London (01) 606 4194
- SYSOP: Dave Coles
- TIMES: 24 hours
- SPEED: 300 Baud, except Wednesdays when 1200/75 Baud
- SIGS: Apple, Atari, IBM, TRS-80

Distel – London (01) 679 1888
- SYSOP: Display Electronics
- TIMES: 24 hours
- SPEED: 300 Baud

Estelle (0279) 443511
- SYSOP: STC Electronic Services Ltd
- TIMES: Office hours
- SPEED: 300 Baud, 1200/75 on (0279) 441188
- NOTES: Commercial; intended for use by STC customers; 1200/75 system uses Viewdata graphics

Forum-80 Hull (0482) 859169
- SYSOP: Fred Brown
- TIMES: 12.00am - 8.00am all week, 5.00pm - 10.00pm Tuesday and Thursday, 1.00pm - 10.00pm weekends
- SPEED: 300 Baud
- NOTES: The first BBS in Britain

Forum-80 London (01) 902 2546
- SYSOP: Victor Saleh
- TIMES: 9.00pm - 12.00am
- SPEED: 300 Baud

Hamnet (0482) 497150
- TIMES: 24 hours
- SPEED: 300 Baud
- NOTES: BBS for Radio Hams

Liverpool Mailbox (051) 4288924
- SYSOP: Peter Toothill
- TIMES: 24 hours
- SPEED: 300 Baud
- SIGS: Adventure, Apple, Atari, BBC, CP/M, Dragon/Coco, IBM PC, Micro-Wave, Modems, TRS-80
- NOTES: Also has area to leave messages to Personal Computer World magazine

Mailbox-80 W. Midlands (0384) 635336
SYSOP: Jim Roden
TIMES: 5.30pm to 8.00am, all day Sunday
SPEED: 300 Baud
SIGS: Apple II, Atari, BBC, Dragon, IBM PC, Osborne, TRS-80, VIC 20

Manchester BBS (061) 4271596
SYSOP: Robert O'Donnell
TIMES: 24 hours
SPEED: 300 Baud
SIGS: BBC, Hackers, Jokes, Problems
NOTES: First BBS to be run on a BBC micro with the software now available from Pace

Manchester Open BB (061) 7368449
SYSOP: Ken Farnen
TIMES: 24 hours
SPEED: 300 Baud
SIGS: Acorn, Adventures, Apple, Atari, Commodore, CP/M, For Sale/Wanted, Gossip, Hardware, Help!, Jokes, Languages, Modems, Spectrum, Tandy

Maptel (0702) 552941
SYSOP: Maplin Electronic Supplies
TIMES: 24 hours
SPEED: 300 Baud
NOTES: Commercial BBS containing details of stock levels; customers may order supplies with a credit card

Metro BBS London – (01) 341 7840
SYSOP: Paul Beaumont
TIMES: 24 hours
SPEED: 1200/75 Baud
SIGS: Adventure, Art Gallery

Micro Live BBS (01) 579 2288
TIMES: 24 hours
SPEED: 300 Baud
SIGS: MicroLive
NOTES: BBC 'Micro Live' programme BBS; programme notes available for downloading

Microweb (061) 4564157
SYSOP: Mike Bibby
TIMES: 24 hours
SPEED: 300 Baud
SIGS: BBC, News
NOTES: Micro User magazine's BBS

N. Birmingham BBS (0827) 288810
SYSOP: Paul Smith
TIMES: 24 hours
SPEED: 300 Baud
SIGS: Atari, BBC, Chain Letters, Hackers, Infocom, Lonely Hearts

OSI/Technical BBS
SYSOP: Frank Leonhardt
TIMES: 24 hours
SPEED: 300 Baud
NOTES: Ring Back, bias toward computer projects, home-brew software under constant development!

PIP (0742) 667983
SYSOP: Quentin Ridford
TIMES: 24 hours; 9.00am - 0.00am CCITT, 0.00am - 9.00am BELL
SPEED: 300 Baud
SIGS: Adventure Clues, Apple, Apple 'Mac', BBC, CP/M, DOS, Fantasy Area, IBM PC, Modem Spot, MUD, Oric, Sales and Wants

Southern BBS (0243) 511077
SYSOP: Jonathan Sanders
TIMES: 24 hours
SPEED: 300 Baud
SIGS: Atari, BBC, Bulletins, For Sale, Hackers, Zork

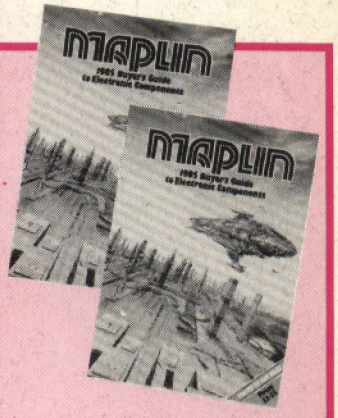
Stoke ITec (0782) 265078
SYSOP: Ian Hickman
TIMES: 24 hours
SPEED: 300 Baud
NOTES: Remote CP/M system

TBBS Blanford (0258) 54494
SYSOP: Leo Knaggs
TIMES: 24 hours
SPEED: 300 Baud
SIGS: Apple, Atari, BBC, Commodore, Modems, Newbrain, TRS-80

TBBS London (01) 348 9400
SYSOP: John Newgas
TIMES: 24 hours
SPEED: 300 Baud
SIGS: Apple, BBC, C, Forth & Pascal, Chain Letters, CP/M, Diplomacy, Graffiti, Hackers Club, Modem Spot, MS-DOS, Politics, Radio Comms, Sales and Wants, Where to eat
NOTES: Download, (games, adventure hints)

All new in the 1985 Catalogue

From a gentle purr to a mighty roar, the tightly controlled power of the beast is yours to command!



PROFESSIONAL QUALITY HIGH POWER LOUDSPEAKERS

A new range of superb quality loudspeakers.

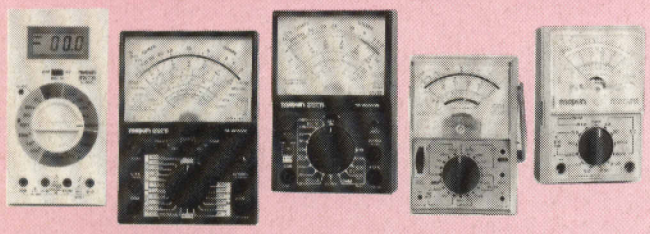
- ***Virtually indestructible high temperature voice-coil reinforced with glass-fibre***
- ***100% heat overload tolerance***
- ***Advanced technology magnet system***
- ***Rigid cast alloy chassis***
- ***Linen or Plastiflex elastomer surrounds***
- ***5-year guarantee (in addition to statutory rights)***

Available in 5, 8, 10, 12, 15 and 18 inch models with 8Ω and some 16Ω impedances and with input powers ranging from 50W to 300W e.g.

5in. 50W 95dB 8Ω: XG39N / 16Ω: XG40T £17.95§
8in. 100W 98dB 8Ω: XG43W £29.95§
10in. 100W 100dB 8Ω: XG46A £29.95§
12in. 100W 101dB 8Ω: XG49D £29.95§
12in. Twin Cone 100W 100dB 8Ω: XG50E / 16Ω: XG51F £31.95§

Note - the output power doubles for each 3dB increase (ref 1W @ 1m).

PRECISION GOLD MULTIMETERS

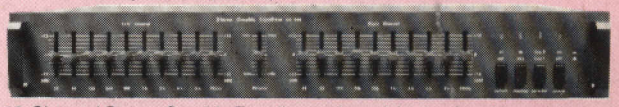


A new range of very high quality multimeters offering truly amazing quality at the price.

Pocket Multimeter, 16 ranges, 2000Ω/V DC/AC £6.95§ (YJ06G)
M-102BZ with Continuity buzzer, battery tester and 10A DC range, 23 ranges, 20,000Ω/V DC £14.95§ (YJ07H)
M-2020S with Transistor, Diode & LED tester and 10A DC range, 27 ranges 20,000Ω/V DC £19.95§ (YJ08J)
M-5050E Electronic Multimeter with very high impedance, FET input, 53 ranges including peak-to-peak AC, centre-zero and 12A AC/DC ranges £34.95§ (YJ09K)
M-5010 Digital Multimeter with 31 ranges including 20Ω and 20μA DC/AC FSD ranges, continuity buzzer, diode test, and gold-plated PCB for long-term reliability and consistent high accuracy (0.25% +1 digit DCV) £42.50§ (YJ10L)

Our huge range of top quality electronic components at very competitive prices are all detailed in our catalogue, and with well over 600 new lines in our 1985 edition and many design improvements, it's well worth getting a copy. Here are just a few examples from the catalogue. (The items below are NOT kits).

★ Most phono and jack plugs now with integral strain relief sleeve - gold-plated types also available from 14p (gold from 70p)
★ Stereo Disco Mixer with cross-fade, talk-over, cue monitoring, aux input, slide controls. Only £58.95 (AF99H)



★ 10-Channel Stereo Graphic Equalisers - 3 models - basic; with peak level meter; and with spectrum analyser - from £77.95

★ Digital Delay Line permits Slap-back, Doubling, Flanging, Chorus and Echo. 11 controls. Only £195.00 (AF98G)
★ Video Enhancer improves picture quality when recording from one VTR to another, and with TV's with monitor input. Only 28.95 (XG59P)
★ Detailed descriptions of the exciting new 74HC range of IC's which combine the advantages of CMOS and TTL. From 46p
★ Keyboards: sloping keys, two-tone grey, mounted in steel frame, very smart cases (extra) available. 61 keys, only £33.95 (YJ12N)
79 keys, only £37.95 (YJ13P)
★ 1% Resistors now 50ppm/°C, 0.4W, only 2p each!
★ Auto transformers 120/240V 50VA, £10.75§ (YJ56L). 100VA £14.95§ (YJ57M). 150VA £16.95§ (YJ58N). 250VA £21.95§ (YJ59P).
★ Digital Clinical Thermometer. Only £13.95 (FK51F)

Check our 1985 Catalogue for all our other fascinating new lines.

1985 CATALOGUE

Pick up a copy now at any branch of W.H. Smith or in one of our shops. The price is still just £1.35, or £1.75 by post from our Rayleigh address (quote CA02C).

AMERICAN EXPRESS · map · Access · VISA

☎ Phone before 2pm for same day despatch.

N.B. All our prices include VAT and Carriage. A 50p handling charge must be added if your total order is less than £5 on mail order (except catalogue).

MAPLIN ELECTRONIC SUPPLIES LTD.

Mail Order: P.O. Box 3, Rayleigh, Essex SS6 8LR. Tel: Southend (0702) 552911
SHOPS
- BIRMINGHAM Lynton Square, Perry Barr, Tel: 021-356 7292.
- LONDON 159-161 King Street, Hammersmith, W6. Tel: 01-748 0926.
- MANCHESTER 8 Oxford Road, Tel: 061-236 0281.
- SOUTHAMPTON 46-48 Bevois Valley Road, Tel: 0703 25831.
- SOUTHEND 282-284 London Rd, Westcliff-on-Sea, Essex. Tel: 0702-554000

Shops closed all day Monday.

Post this coupon now for your copy of the 1985 catalogue. Price £1.35 + 40p post and packing. If you live outside the U.K. send £2.40 or 11 International Reply Coupons. I enclose £1.75.

Name .....
Address .....
.....
.....

E&CM 3/85

§ Indicates that a lower price is available in our shops. ***All offers subject to availability.*** ***Prices firm until May 11 1985.***